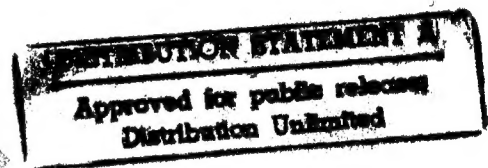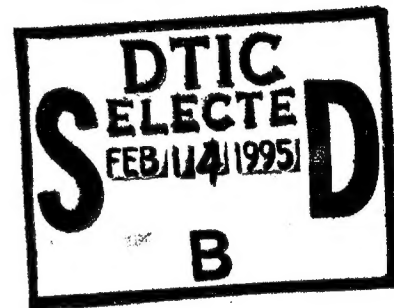Carnegie Mellon University
Software Engineering Institute

Software Risk Evaluation Method
Version 1.0

Frank J. Sisti
Sujoe Joseph

December 1994

DTIC
SELECTED
FEB 14 1995
B

19950207 009

Software Risk Evaluation Method
Version 1.0

Frank J. Sisti

Sujoe Joseph

Enterprise Risk Management Project

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

This report was prepared for the

SEI Joint Program Office
HQ ESC/ENS
5 Eglin Street
Hanscom AFB, MA 01731-2116

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

Review and Approval

This report has been reviewed and is approved for publication.

FOR THE COMMANDER

Thomas R. Miller, Lt Col, USAF
SEI Joint Program Office

Preface

Historically decision makers have had to manage the risks of software-intensive programs and projects by applying heuristic methods based on their own individual skills and experience. As software-intensive systems increase in complexity and size, however, it is essential for decision-makers to use more disciplined and systematic methods for managing software risks.

By using defined methods that are suited to the state of the practice, decision makers obtain the insights necessary to take actions that are often crucial to the success of their programs and projects.

To effectively manage software risks, it is necessary to use defined methods in many areas of risk management, including identification and analysis. It is imperative that the methods be able to facilitate communication among all parties and at all levels of the organization.

In addition, it is necessary that the methods promote the efficient management of software risks and provide decision makers with sufficient information to focus on the priority risks and mitigate them with the scarce resources that are available to them.

The Software Engineering Institute (SEI) has designed and developed the Software Risk Evaluation (SRE) method as one of the methods of software risk management that can meet the specific needs of decision-makers who are responsible for managing software-intensive programs or projects.

One of the goals of the developers is to make the method flexible and adaptable to the practical needs of software-intensive programs and projects. A simultaneous goal is to ensure the method is also well defined and can be applied systematically and in a disciplined manner within any organization.

The SRE method is based on the foundational work that was laid within the Risk Program of the Software Engineering Institute. It is also one of the products that facilitate the implementation of the SEI risk management paradigm.

The robustness of the latest version of the SRE method is a result of continuous application and field testing on actual software programs and projects. This has over time both improved and enhanced the method to the current state of the practice.

The functional components of the SRE method and its implementation phases and activities are designed to be effective for managing the software risks of programs and projects. In addition the integrated tools and techniques of the SRE method enable its utility as an efficient and practical method.

Acknowledgments

The current version of the Software Risk Evaluation method is the culmination of a number of work efforts within the Software Engineering Institute that have occurred over the past few years. It would be remiss if we did not recognize their contributions.

Guidance and vision for the development of the SRE mechanism and this version came from all SEI Risk Program personnel with particular emphasis from Clyde Chittister. Comments provided from outside the SEI came principally from members of the Software Action Plan Working Group of the Office of the Deputy Defense Research and Engineering. The chairman of that group, Dr. Barry Boehm, was the initiator of the work, and Ms. Virginia Castor maintained support for the early portion of the effort, which has resulted in the finished product.

The Risk Taxonomy, one of the foundations of the SRE method, is the result of the efforts of Marvin Carr, Suresh Konda, Ira Monarch, Carol Ulrich, and Clay Walker. Without their earlier work (see CMU/SEI-93-TR-6), the creation of the SRE would have taken that much longer and would not have attained the level of robustness that it currently enjoys. Our colleagues in the Risk Program have not only provided meaningful comments during the development of the method, but have also been part of the field efforts over the past year.

To our project team colleagues, a special thanks and recognition. Mike Dedolph, Carol Ulrich, Rob Wojcik, Bill Wood and our resident affiliate Anne Quinn have all helped in the difficult work of testing the method in a number of instances around the country. Lorrie Orndoff deserves mention for providing us the administrative support and hence the document.

Our Organizational Capability Development colleague, John Maher, deserves credit for having the outside insight to assist us in focusing the final aspects of the effort, which have made it that much better than it would have been otherwise.

At the same time, we want all of the individuals who have assisted in the creation of this technical report over the past year and a half to understand that the impact of this method is due in part to their efforts as well. For this we thank you.

The Authors

Table of Contents

List of Figures

List of Tables

Software Risk Evaluation Method

Abstract

The Software Risk Evaluation (SRE) method is used for identifying, analyzing, communicating, and mitigating software risks.

The SRE method is intended to be used by decision-makers for managing the software risks of software-intensive programs and projects. The SRE method facilitates the mitigation of software risks for managers.

This document reports on the Software Risk Evaluation method version 1.0 and provides a high-level description of the method. The report is intended to provide the reader with an overview of the functional components and the implementation aspects of the method.

The current version of the SRE method evolved from two earlier versions. The first version was developed under a technical collaboration agreement with the Director of Defense Research & Engineering (DDR&E) for mitigating the risks inherent in software acquisitions. The latter version was developed to improve and enhance the capabilities of the method for users in software-intensive programs and projects.

1   Introduction

1.1   Purpose and Scope

Purpose

The purpose of this document is to provide a technical report on the Software Risk Evaluation (SRE) method. The report describes both the functional components and the implementation aspects of the SRE method.

This report provides an overview of the SRE processes, activities, tools, and techniques. It also describes some of the previous work done at the Software Engineering Institute (SEI) that was foundational for the development of the SRE method.

The SRE technical report is intended to provide the reader with a general understanding of the SRE method. It does not define the details that are required for the practice of the method. The implementation details are available in other materials such as SRE handbooks, guidebooks, and templates. The practical aspects of the SRE method will be facilitated through a training course offered by the SEI.

Scope

The scope of this document is to answer the following:

- What is the SRE method, and what are its functional components?
- What is the overall process for applying the SRE method in practice?

1.2 Background

DDR&E Needs

In 1991 the Director of Defense Research & Engineering (DDR&E) established a Software Action Plan Working Group (SWAP-WG) to identify the software-related areas that could be leveraged by the timely application of resources.

One of the identified areas was the risks associated with software acquisitions and the need to identify and mitigate the risks at an early stage. The SWAP-WG decided to address this need by using the ongoing work at the Software Engineering Institute (SEI) and by enlisting the expertise that was available within the SEI Risk Program.

Version 0.1

The initial document (version 0.1) of the SRE method was developed based on a technical agreement that was signed between the SEI and the DDR&E in 1993.

The SRE method version 0.1 was reviewed and successfully field tested on two software programs. The field tests identified areas for improvement of the SRE method.

Version 0.2

The SRE method version 0.2 was produced in early 1994. It documented the improvements that were made to the earlier version. The documentation included modifications that improved the readability as well as implementation efficiency of the method.

The SRE method version 0.2 has been field tested on many software-intensive programs and projects. With each field test, one aspect or another of the method has been improved or enhanced.

The feedback from individuals based on their review of version 0.2 documentation also enabled many improvements, particularly to the structure of the document.

Version 1.0

This report contains the enhancements and improvements that were made to the previous SRE version 0.2 document.

The enhancements to the previous version are mainly in the SRE commitment process, that is, the steps required prior to executing the method, in the risk mitigation functions, and in associated implementation.

The improvements to the previous version are a result of the field testing and validation that were done on the SRE method application on a variety of software-intensive programs and projects. This report also includes improved terminology and easy-to-understand vocabulary as suggested by the reviewers of the previous document (version 0.2).

1.3 Strategic Direction

Long Term Goals

While the immediate focus of the development team is to develop and continuously improve the SRE method, the long-term goals are to ensure that

- The method is defined and can be applied in a systematic, disciplined, and efficient manner.
- For a specific program or project, at any given instance of its application, there is uniformity in the outcome of the risk findings and mitigation.
- It is flexible enough to be used in different situations and phases of the life cycle, including acquisition and maintenance.

Duality in Approach

To meet the long-term goals, the SEI has adopted a dual and evolutionary path of development for the SRE method. The first path concerns the development of the method itself in terms of its processes, techniques, and tools, and is documented in this report.

The second path concerns the capture of knowledge from various domain experts to build a predictive decision model for proactive management of software risks.

Evolutionary Development

The evolutionary development cycle of improvements, field testing, and validation enables the SRE method to become comprehensive and defined.

The evolutionary development provides the required maturity in the operational details and ensures existing techniques and tools are enhanced and new ones are added to meet the practitioner's needs.

Predictive Decision Model

The development of a predictive decision model for software risks is of strategic interest to the SEI and the user community. This challenging task requires the employment and use of knowledge engineering techniques.

The premise of the predictive decision model as stated in Morgan [Morgan 81] and others is that although good risk data may not be available for a particular program or project, the data available from other similar program or project experiences may be adapted or extended for use either directly or as part of a general model for predicting risks and their associated mitigation efforts.

Knowledge Domains

Knowledge domains of the predictive decision model pertain to areas such as the characteristics of a software-intensive program or project, typical structures of software risks and their drivers, interactions among software risks and their cause-and-effect relationships.

Knowledge also pertains to domains that are required for the understanding and capture of the cognitive and intellectual means that people use to process and communicate risks.

The development of a predictive decision model for risk management requires the use of knowledge engineering principles and techniques such as protocol analysis [Poltrock 89].

2   Technical Basis

2.1   Risk Management

Paradigm

The SEI risk management paradigm defines a continuous set of activities that must be undertaken to identify, communicate, and resolve software risks.

The model used to represent this paradigm is a circular set of activities emphasizing that risk management is a continuous process (see Figure 2-1).

The arrows within the model represent the logical path and the sequence in flow of information within the risk management paradigm.
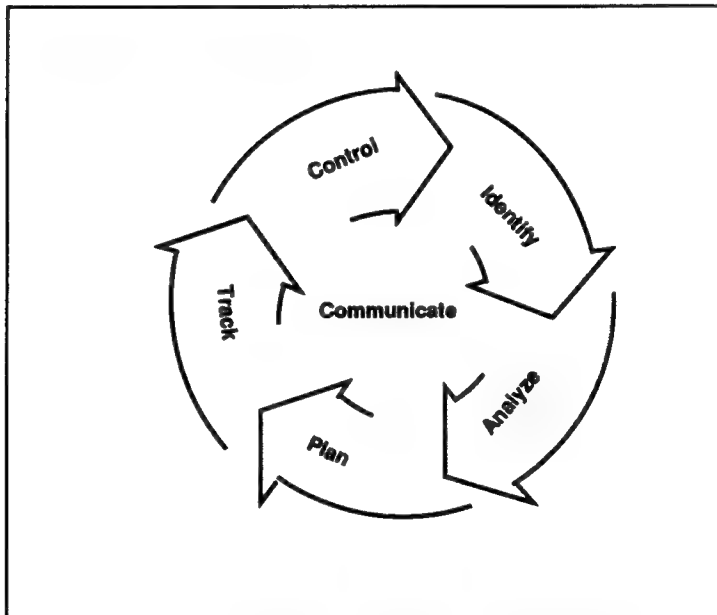
Figure 2-1: Risk Management Paradigm

Activities

The activities in the software risk management paradigm are:

Identify: Surfacing software-related risks before they become problems that can adversely affect the program or project.

Analyze: Transforming data from the identified risks into decision making information.

Plan: Using the risk information in decisions and actions including developing actions to mitigate individual risks, prioritizing actions, and integrating them into an executable risk management plan.

Track: Monitoring the status of risks and their mitigation actions along with the use of metrics and triggering events.

Control: Correcting the deviations from planned risk mitigation actions by using existing program or project management control functions.

Communicate: Exchanging risk management information among the functions and at all levels of the organization. This activity is represented in the center of the model to emphasize its pervasiveness and criticality for implementing the other activities in the paradigm.

2.2 Software Risk Taxonomy

Risk Taxonomy

The SEI software risk taxonomy provides a basis for organizing and studying various aspects of software risks in a program or project. It serves not only as a systematic way of eliciting and organizing risks but also as a consistent framework for the development of risk management methods and techniques. The taxonomy has been developed over the past three years and has been validated in over 30 instances.

Classes

The taxonomy is organized into three major *classes* as follows:

Product Engineering: covers the technical aspects of the work to be accomplished.

Development Environment: covers the methods, procedures, and tools used in the production of the software product.

Program Constraints: covers the contractual, organizational, and operational factors within which the software must be developed and that are generally not under the direct control of the organization.

Elements

The classes in the taxonomy are divided into *elements*; for example, the Product Engineering class is divided into Requirements, Design, Code & Unit Test, Integration and Test, and Engineering Specialties elements.

Attributes   Each element is further divided into *attributes*; for example, the Requirements element is divided into the attributes of Stability, Completeness, Clarity, Validity, Familiarity, Precedent, and Scale.

Taxonomy Structure Figure 2-2 shows the structure of the software risk taxonomy.
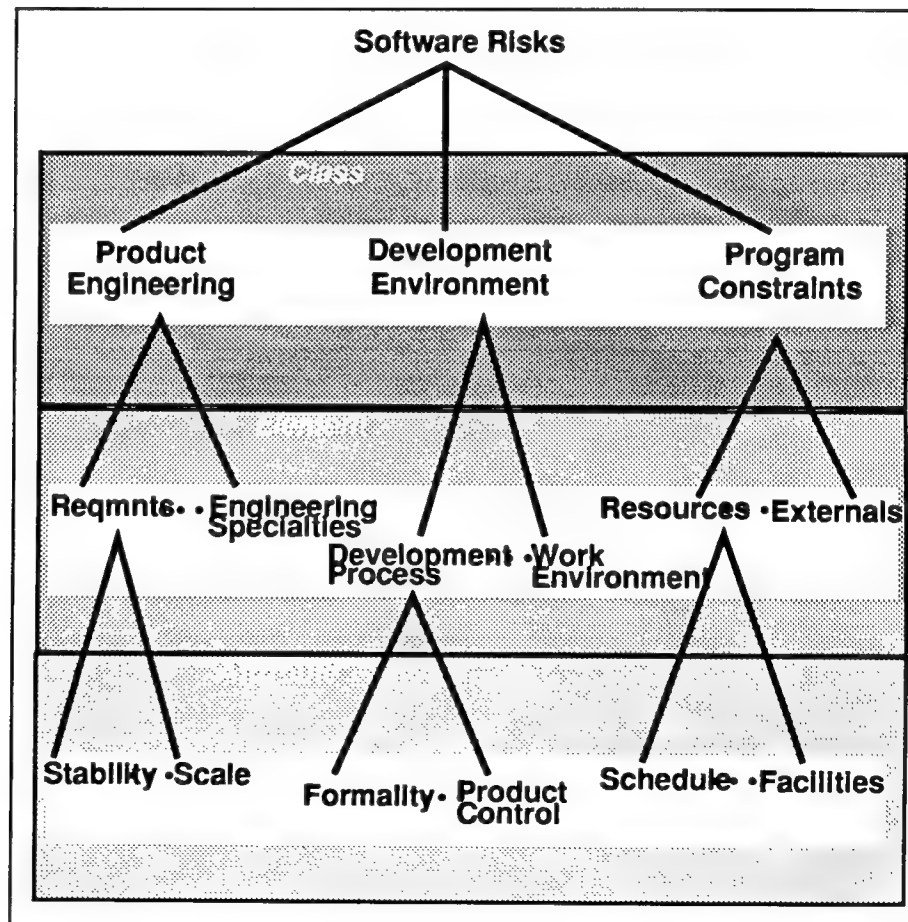
Figure 2-2: Structure of the Software Risk Taxonomy

2.3 Taxonomy-Based Questionnaire

Purpose

The Taxonomy-Based Questionnaire (TBQ) is a tool used to identify software risks in a program or project.

The purpose of this tool is to ensure coverage of all potential risk areas by asking questions at a detailed attribute level of the software risk taxonomy.

Typical Usage

The TBQ contains specific questions to find risks, and follow-up questions and cues that enable the person administering the questionnaire to probe for risks.

The TBQ is effective when used along with appropriate interview techniques and when applied on management and technical groups within an organization. The TBQ is also found to be more effective when administered by an independent team on peer groups.

Since the TBQ is a general tool, some of its questions may not be applicable to specific programs or projects. The TBQ is therefore tailored to suit the specific needs of an organization before it is administered.

TBQ Conventions   The TBQ uses specific conventions to facilitate effective and consistent administering of the questions. For example, a question that is framed within square brackets (see figure 2-3) is used as a "starter" to provide a generalized context for the respondents. Depending on the initial response, each question may have additional probe questions.

A.   Product Engineering

1.   Requirements

a.   <u>Stability</u>

[Are requirements changing even as the product is being produced?]

[1] Are the requirements stable?
(No)   (1.a)   What is the effect on the system?
- Quality
- Functionality
- Schedule
- Integration
- Design
- Testing

[2]   Are the external interfaces changing?

Figure 2-3: Taxonomy-Based Questionnaire View

The probe questions are prefaced in parentheses with "Yes" or "No," indicating the type of initial response that should activate its use during the interview.

Some questions may also have "cues" (or a list of items in bullet format). These cues are used to trigger thoughts in the minds of the respondents on issues, concerns, or risks.

3 Applications

Introduction

The SRE method is designed to be a flexible method that can be suitably applied in a variety of situations both in a development environment and in an acquisition environment. The SRE method can also be suitably applied during any phase of the development or acquisition life cycle.

Sponsor

Successful application of the method depends on the availability of sponsorship. The sponsor ensures that the goals for applying the SRE method are attained. Typically the sponsor of a software risk evaluation is one of the following:
- program or project manager
- acquisition manager
- corporate manager
- individual change agent

3.1 System Environment

System View

The SRE method views risks from a system level. It deals with the many uncertainties associated with the development and acquisition of complex software-intensive programs and projects.

Risk Areas

Representative areas of risk in a system environment are software, hardware, technology, cost and schedule, and people. The risks in these areas have complex interactions and interdependencies with each other.

Figure 3-1 shows the risk areas and their inter-relationship in a system environment.
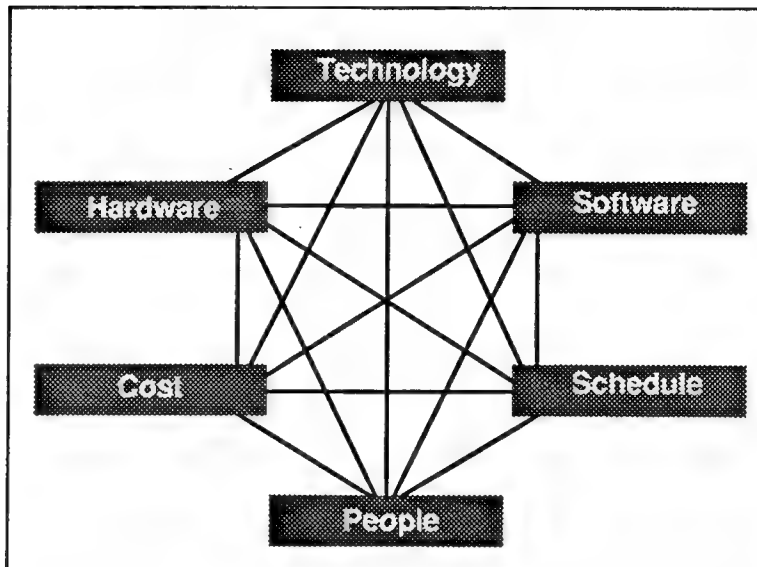
Figure 3-1: Risks Within a System Environment

3.2 Sample Application

Application

The SRE can be applied in different situations and within different environments. One example of an SRE application is shown in Figure 3-2. Here the SRE method is applied in a program where a government buyer-contractor supplier relationship exists.

In this scenario, the program manager or client recruits an independent team to perform an SRE. The independent team executes the SRE functions and provides a data confirmation briefing and feedback to the contractor organization.

Executing the SRE produces data pertaining to risks the independent team analyzes within the context of program or project goals and environment. By working together with the representatives from the respective organizations, the SRE team develops mitigation strategies and specific activities for managing the risks.

The results of the risk evaluation are then provided to the client in a final report. The final report contains both the risk findings and suggestions to mitigate them.

Note that the application of the SRE method is not limited to this scenario. The SRE method can be applied in a variety of situations, for example, the software developer or contractor can apply the method internally as a tool for managing software risks.
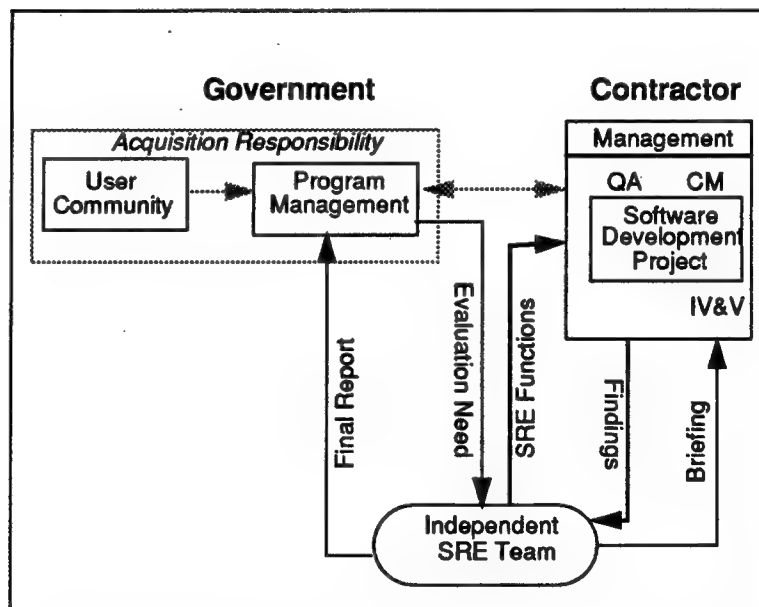
Figure 3-2: Sample Application

3.3 Notional Timeline

Timeline

The notional timeline for conducting a software risk evaluation is shown in Figure 3-3.
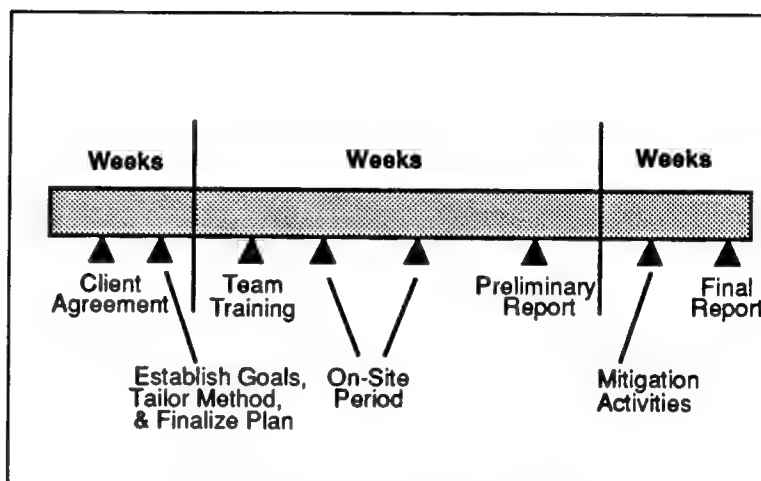
Figure 3-3: Notional Timeline

For the sample application that was described in the previous section, the approximate schedule is as follows:

- Client Agreement   2 or 3 weeks
- Establish Goals, Tailor
  Method, and Finalize Plan 1 or 2 weeks
- Team Training   2 days
- On-Site Period   1 week
- Preliminary Report   3 or 4 weeks
- Mitigation Activities   1 week
- Deliver Final Report   1 or 2 weeks

4 Functional Components

4.1 Overview

Function Types

The Software Risk Evaluation method consists of two types of functions – primary and support functions (see Figure 4-1).

The primary functions are Detection, Specification, Assessment, Consolidation, and Mitigation. The support functions are Commitment, Planning & Coordination, Verification & Validation, and Training & Communication.
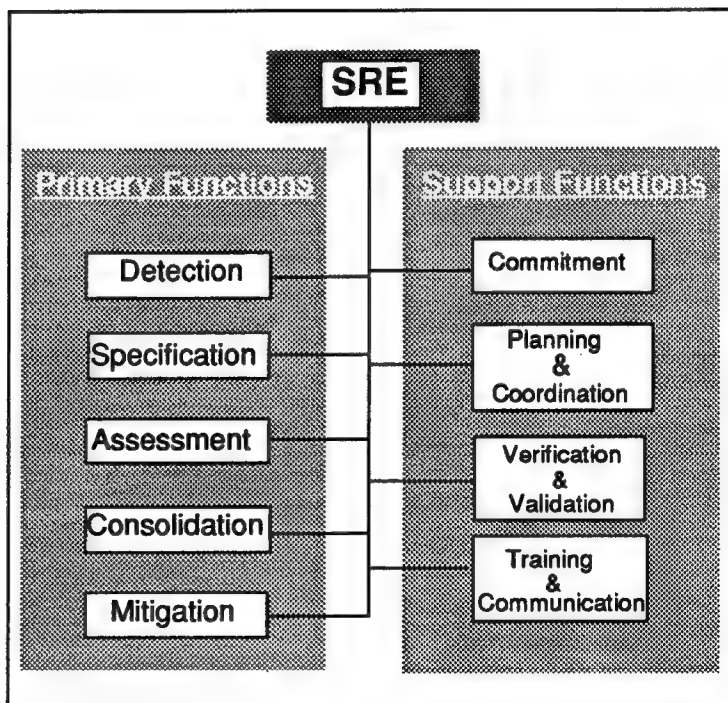
Figure 4-1: Functional Components

4.2   Primary Functions

4.2.1 Detection

Purpose

Detection is the function of finding the software risks associated with a software-intensive program or project.

This function is performed to ensure systematic and broad coverage of all potential risk areas. It is also performed to ensure efficiency and effectiveness in finding software risks through the use of appropriate tools and techniques.

Key Items

Risk detection in the SRE method is performed by using the following key items:

- SEI Taxonomy-Based Questionnaire to ensure complete coverage of all areas of potential software risks.
- An independent team conducting peer group interviews using the specified process and techniques.
- Selection of the peer groups and the appropriate individuals within each group using the SRE guidelines.
- Training of the independent SRE team in the SRE method and the effective use of its tools and techniques.

4.2.2 Specification

Purpose

Risk specification is the function of recording all aspects of the identified software risks including its conditions, consequences, and immediate source of the risk for taking corrective actions.

Representations

There are many ways of representing software risks that range from simple statement of the issue to a comprehensive specification statement.

Two representations are provided as examples below. Both have been field tested and found to be effective in practice.

Representation-1

Figure 4-2 shows an example of a simple statement of issue with the software risk implied.

> *Changes to the logical database design during coding phaseare causing changes to program modules that access the database.*

Figure 4-2: Simple Representation

Representation-2 Gluch [Gluch 93] has developed a structured representation for software risk statements. The syntactical construction and an illustration of its use are shown in Figure 4-3.

The purpose of the structured representation is to:
- Serve as the guiding structure to identify and communicate risks.
- Capture the components of the risk and to some degree simplify the task of prioritizing.
- Isolate and record the condition when the risk becomes valid.
- Assist in the identification of source(s) of the risk.
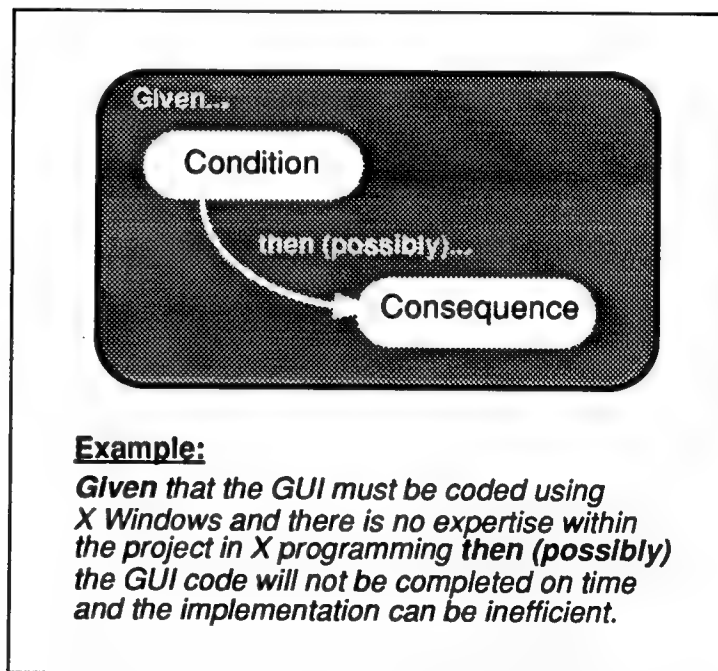
Figure 4-3: Structured Representation

Source of Risk

Risk specification also includes identifying the source of the risk.

In the SRE method each risk is assigned to a specific category within the SEI taxonomy of software risks.

The source of each risk is specified by tagging it to a class, element, and attribute of the taxonomy. For example, if the immediate source of risk is determined to be unstable requirements it is tagged as "A1a" (see Figure A-1 for taxonomy groups).

4.2.3 Assessment

Purpose

Risk assessment is a function that is performed to determine the magnitude of each software risk. The primary purpose of the function in the SRE method is to prioritize the risks and to effectively mitigate them.

Definition

By definition, the magnitude of a risk is a function of its severity of impact and the probability of its occurrence (see Figure 4-4).

Risk Magnitude = Severity of Impact * Probability of Occurrence

Figure 4-4: Definition of Risk Magnitude

Mechanisms

There are different mechanisms that can be used for assessing risk magnitude ranging from simple subjective measures to more rigorous statistical measures.

In the current version of the SRE method, two mechanisms are provided as examples (see below). Both are found to be practical and suitable for performing the risk assessment function.

Mechanism-1

The first assessment mechanism uses a simple scale where risk statements are assessed at one of four levels of magnitude depending on their impact on the following factors:

• Cost and schedule
• Performance
• Supportability

Table 4-1 lists the four levels of magnitude and their suggested guidelines.

Table 4.1: Levels of Magnitude and Guidelines

Magnitude	Guidelines
Critical	• High likelihood of the risk severely impacting one or more factors i.e., cost & schedule, performance, and supportability
High	• High likelihood of the risk moderately impacting one or more factors
Medium	• Medium likelihood of the risk moderately impacting one or more factors
Low	• Low likelihood of the risk moderately impacting one of more factors

The rationale for using this mechanism is as follows:

- To keep the scoring mechanism as simple as possible.
- To use a subjective scoring mechanism that can be effective for prioritizing the risks.
- To surface the differences in opinion among individual team members regarding the magnitude of a risk and to reach consensus.
- To form a basis for conducting the technical consensus discussions during the analysis sessions.

Mechanism-2

A more involved mechanism for risk assessment is one which is adapted from the Air Force [Airforce 88] and modified for use in the SRE method.

In this mechanism, risk statements are assessed at one of three levels of magnitude:

- High
- Medium
- Low

The level of magnitude of a particular risk depends on the assessment of its severity of impact and its probability of occurrence.

Severity of impact is the effect of the particular risk on the target program or project and is determined on the basis of its impact on the software's performance, supportability, cost, and schedule.

Each risk is determined to be at one of the following levels of severity:

- Catastrophic
- Critical
- Marginal

Probability of occurrence is the certainty or likelihood of the risk becoming true for the program or project.

Each risk is determined to be at one of the following levels of probability:

- Very Likely
- Probable
- Improbable

The matrix shown in Figure 4-5 is used to determine the level of magnitude based on assessment of the severity of impact and the probability of occurrence of a risk. The shaded areas indicate different levels of magnitude.

For example, a high magnitude risk is one whose severity is catastrophic and probability of occurrence is very likely or probable, or whose severity is critical and probability of occurrence is very likely.

Probability / Severity	Very Likely	Probable	Improbable
Catastrophic	High		
Critical		Medium	
Marginal			Low

Figure 4-5: Risk Magnitude Matrix

4.2.4 Consolidation

Purpose

This function is performed to integrate the multiple risk detection activities of the SRE method both when similar risks are identified from different sessions and when multiple risk evaluations are held for a program or project.

For example, related risks may be identified from different interview sessions during a site visit. Similarly, related risks may be identified from multiple risk evaluations that are performed for a program or project or when separate site visits are done for executing the SRE for the program office and the subcontractors

who are responsible for developing the system or its components.

Criteria

The risk statements that are considered for consolidation must meet *one* of the following criteria:

- Manifestation of the same risk statement, that is, identical in every way except in the wording of the risk statements.
- Fragmentation due to minor variations or different aspects of the same risk statement.
- Differences in granularity, for example, a minor risk statement that is covered in the context of another risk statement of larger magnitude.

Guidelines

Some of the suggested guidelines for risk consolidation activities are listed below:

- Classify related risks in some way, such as by the sources of risk to be more easily identifiable.
- Ensure the context of the risk is maintained by keeping the original wording of the affected risk statements.
- Merge identical risk statements into a single statement.
- Combine fragmented risk statements into a single statement whenever possible.
- Abstract granular risks into a single statement that provides sufficient level of detail for effective risk mitigation.

4.2.5 Mitigation

Purpose

Risk mitigation is the function of developing strategies and specific activities to alleviate and eliminate the threat posed by risks to the success of the program or project.

The mitigation function is performed by identifying areas where the program or project can focus its resources for effectively addressing the risks. These areas are termed mitigation areas.

Risk mitigation develops the strategies and activities with reference to the goals of the program or project for performing the risk evaluation.

Mitigation Area

A risk mitigation area is a logical grouping of similar risks to enable them to be managed efficiently and effectively.
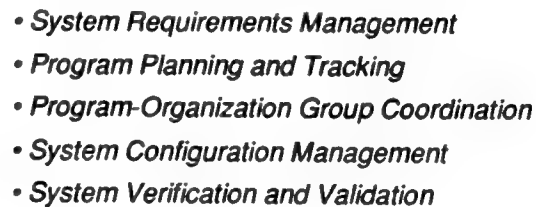
The use of risk mitigation areas is a management technique that allows development and implementation of mitigation activities within a broader perspective of program or project goals and objectives.

The use of risk mitigation areas has been found to be practical particularly when dealing with a large number of risks.

Figure 4-6 is an example of the mitigation areas that were identified for a large-scale integrated database system.

> • System Requirements Management
>
> • Program Planning and Tracking
>
> • Program-Organization Group Coordination
>
> • System Configuration Management
>
> • System Verification and Validation

Figure 4-6: Sample Mitigation Areas

Risk Map

A risk map is an important tool that is used for the development of mitigation strategies and specific mitigation activities for each mitigation area.

The risk mapping exercise is performed by mapping the specific risk findings that were identified to the respective mitigation areas. The risk map should also map the risk findings to the goals of the program or project for performing the risk evaluation.

The risk map ensures that all of the risks are being addressed and that no risk is omitted inadvertently. It also ensures that risk mitigation will address client goals.

Mitigation Stages   The mitigation function is performed in four stages for each mitigation area of the program or project.

Table 4-2 lists the mitigation stages and their descriptions.

Table 4.2: Development Stages for Mitigation Activities

Stage	Description
Analysis	• What is the current status? A description of the current situation of the program or project within each mitigation area is developed. The risk map is used to ensure all risks are taken into consideration.
Goals	• What is the target status? Specific goals are identified for each mitigation area. The goals collectively describe the desired status for the mitigation area. The goals support the program or project goals and objectives.
Strategies	• What can be done to reach the target status? A strategy is developed to reach the desired status (goals) for the mitigation area. The strategy is developed as a set of general or broad recommendations that are candidates for implementation within the program or project. The general recommendations that are to be implemented within the program or project are selected and prioritized by the client management.
Activities	• What activities should be implemented? Each general recommendation that is selected for implementation is developed into specific activities. **Key Activities**: Key activities to implement the recommendation are first identified. If there are any constraints within the program or project, the activities to overcome the constraints are also identified. **Enabling Activities**: Enabling activities such as kick-off meetings, training, developing task plans, support needs, etc., are also identified for implementation.

4.3 Support Functions

4.3.1 Commitment

Purpose

The commitment function is very important for the successful performance of the software risk evaluation and is comprised of two aspects.

The first aspect is to establish a business relationship with the management of the program or project and to understand and establish the goals for performing the software risk evaluation. The second aspect is to gain the commitment in the constraint of how the SRE application will satisfy client program or project goals.

Mechanism

Included in this function are activities that can provide executives with an understanding of the SRE method in terms of its purpose, scope, required resources, execution phase activities, mitigation activities, and typical outcome of a risk evaluation.

Also included are activities for the SRE team to understand the program or project goals and to map those to the activities of the software risk evaluation.

4.3.2 Planning & Coordination

Purpose

Planning and coordination is a supporting function of the SRE method. Its purpose is to prepare for the implementation phases of the evaluation.

Roles

Both the SRE team and the target organization are involved in the planning and coordination function.

SRE team leader is an experienced person designated from an independent organization and responsible for the overall planning and coordination of the SRE implementation activities.

Site coordinator is designated from the target organization to be the single interface for the SRE team leader to perform planning and coordination activities including scheduling individuals for interviews and arranging facilities for the site visit.

4.3.3 Verification & Validation

Purpose

Verification and validation is a supporting function of the SRE method.

Its purpose is to ensure quality of the implementation process and the accuracy and validity of the results. Its purpose is also to ensure reliable information for the effective mitigation of software risks.

Mechanism

Included in this function are activities, tools, and techniques that facilitate corrective actions from the early stages of implementation.

For example, techniques such as risk play-back and team reviews ensure the accuracy of the content, structure, attributes, and context of each risk.

4.3.4 Training & Communication

Purpose

Training and communication is a supporting function of the SRE method.

Its purpose is to ensure effectiveness of the implementation process by ensuring all the people involved have sufficient knowledge, understanding, and skills. Its purpose is also to create an environment for effective dialog and exchange of information during the implementation.

Mechanism

Included in this function are items such as training of SRE team members who will be involved in the software risk evaluation, orientation of management and technical personnel associated with the program or project, and type and contents of briefings.

Also included are specific activities, tools, and techniques that are used during the implementation process to ensure proper training and communication. For example, use of an introduction script that is read or paraphrased by the interviewer at the beginning of each interview session ensures that a proper environment is set for the open communication of risks where the respondents are able to freely discuss any issue or concern.

5 Implementation Phases

Phases

The software risk evaluation is implemented in four phases -- commitment, preparation, execution, and mitigation.

Commitment consists of activities to establish the need, identify program or project goals, and obtain agreements for the software risk evaluation.

Preparation consists of planning and coordination activities that are performed prior to the site visit for executing the software risk evaluation.

Execution consists of activities to implement the SRE functions during a site visit to the location of the target program or project.

Mitigation consists of activities to mitigate the risks that were identified during the software risk evaluation.

Figure 5-1 shows an overview of the implementation phases.
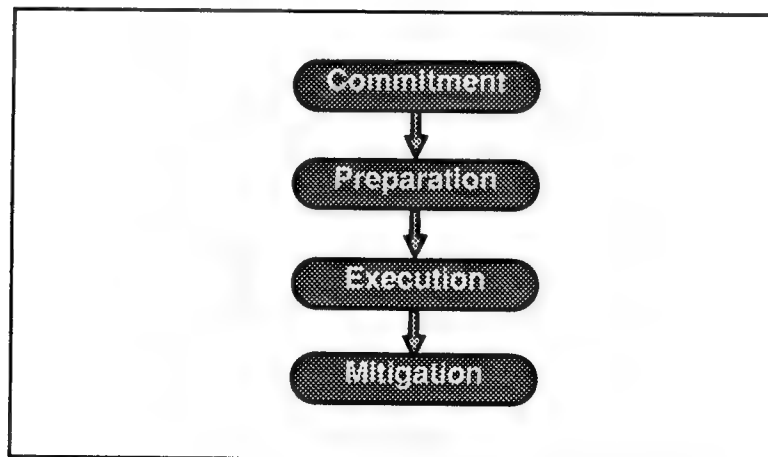
Figure 5-1: Implementation Phases Overview

5.1 Commitment

Introduction

The commitment phase requires at least two meetings and the completion of agreements with the client for performing the risk evaluation.

Figure 5-2 shows the sequence of activities in this phase.
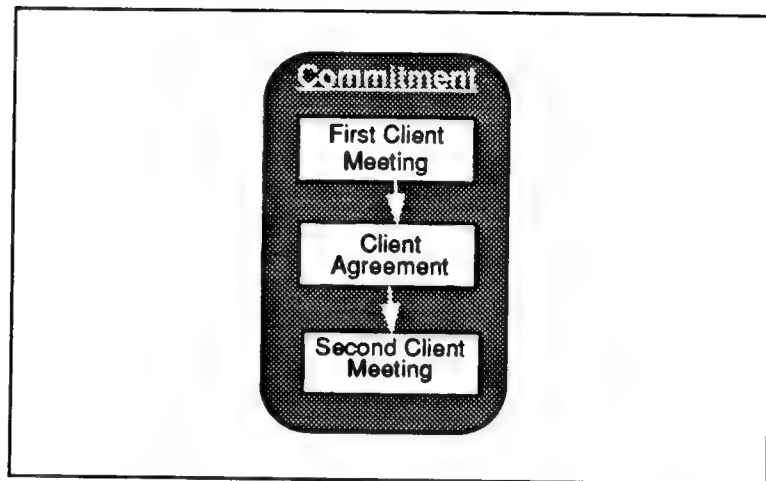
Figure 5-2: Commitment Phase Activities

5.1.1 First Client Meeting

Introduction

The Software Risk Evaluation process begins with the client recognizing the need to determine the risks of a software-intensive program or project. For example, a manager may want to determine the risks of the software development effort to make a critical decision or as a routine activity in the practice of risk management.

Participants

Usually at the first meeting the client's senior management and designees participate in the discussions and provide insight into the established program or project goals.

Participants at the first meeting also include the SRE team leader and other team representatives.

Focus

The focus is to understand the client program or project goals and to determine whether the SRE can meet those goals.

Table 5-1 is a set of suggested topics and outcomes for the first client meeting.

Table 5.1: Focus of First Client Meeting

Topic	Outcome
Introductions	• Participants are introduced and understand each other's functions. • People know what the agenda for the meeting is.
Briefing	• Client has good understanding of the SRE purpose, scope, products, services, and research. • Client begins to formulate a context for the SRE.
Discussion: Client Context	SRE team has good understanding of the strategic context for the evaluation: • What strategic issues are facing the organization? • What are the long-term objectives associated with the completion of the program? • What outcomes are expected from the software risk evaluation?

Table 5.1: Focus of First Client Meeting (continued)

Topic	Outcome
Discussion: Client Goals	SRE team understands the client's goals for the effort: • What are the short-term objectives for the organization? • How critical is the need? • What risks are foreseen for the program from the managers' perspective? • What are the time frames? • What kind of actions is management prepared to take?
Business Relationship	• SRE team and client understand the feasibility of performing a software risk evaluation. • Both parties are ready to do business. • The SRE team understands the risk evaluation's purpose in the context of the client's needs and goals. • Need is determined for linkage of client's (sponsor/-manager) goals with senior management.
Closing	• Outline of next steps to be taken by the respective individuals. • Summary of meeting and action items. • Closing of meeting.

Follow-Up Action   The SRE team leader must follow-up to ensure that the business aspect of the relationship between the client and the SRE team is established at this point.

5.1.2 Client Agreement

Documenting Agreements

One of the techniques to clarify the agreement for the risk evaluation and to ensure its understanding by the client and the SRE team is to document the agreement and have both parties sign it. For example, the SEI uses a Technical Objectives and Plan (TO&P) document that is formally approved by both the SEI management and client management as the basis for conducting the risk evaluations.

The client program or project goals are documented and the goals that are identified are mapped to the software risk taxonomy. This activity is necessary to focus the software risk evaluation on the goals.

Target Project Acceptance

It is also important to obtain proper acceptance from the management and other executives of the program or project. The SRE team leader should ensure that the client has communicated the decision to perform the risk evaluation to the proper executives and that they have been informed well in advance.

5.1.3 Second Client Meeting

Purpose

The purpose of the second meeting is to obtain closure and agreements for the Software Risk Evaluation.

In addition to the individuals from the first meeting, technical focus group(s) from the client organization may also participate at this meeting.

Focus

The focus of this meeting is to address any issues that the client may have and to review the already established client program or project goals for the Software Risk Evaluation.

This meeting should also address the logistics and specific resources that are required for the evaluation.

Table 5-2 is a set of suggested topics and outcomes for the second client meeting.

Table 5.2: Focus of Second Client Meeting

Topic	Outcome
Introductions and Meeting Context	• Some comfort level and relationships are established. • Purpose of meeting – understanding the SRE, client's profile, context for the SRE, and management frame of reference. • Context for modifying the SRE processes including tailoring to meet the client's goals.
Review of SRE	• Client understands what the SRE does, how long it takes, what resources are required, roles, and the typical outcome after performing an SRE. • Client understands the confidentiality of the data.
Discussion: Client Issues	• SRE team understands the client's concerns related to risks of the program or project. • SRE team addresses the client's concerns regarding the risk evaluation.

Table 5.2: Focus of Second Client Meeting (continued)

Topic	Outcome
Discussion: Client Goals and Context of SRE	• SRE team understands what the client wants from the evaluation. • SRE understands the client's context in terms of the organization's strategic direction and goals.
Logistics of SRE	• Client and SRE team agree on logistics for the software risk evaluation: dates, resources needed, site coordinator, etc. • Client understands the importance of the site coordinator role.
Summary and Agreements	• Outstanding issues related to the conduct or purpose of the SRE are addressed or identified. • Agreements made during the meeting, dates for the evaluation, site coordinator, etc., are summarized. • Date, time, and personnel identified for the next step are established.

5.2 Preparation

Introduction

The preparation phase consists of tailoring the risk evaluation, selecting the SRE team and interview group, and scheduling of activities. The preparation activities are performed by the SRE team leader and the site coordinator.

Figure 5-3 shows the sequence of activities for the preparation phase.
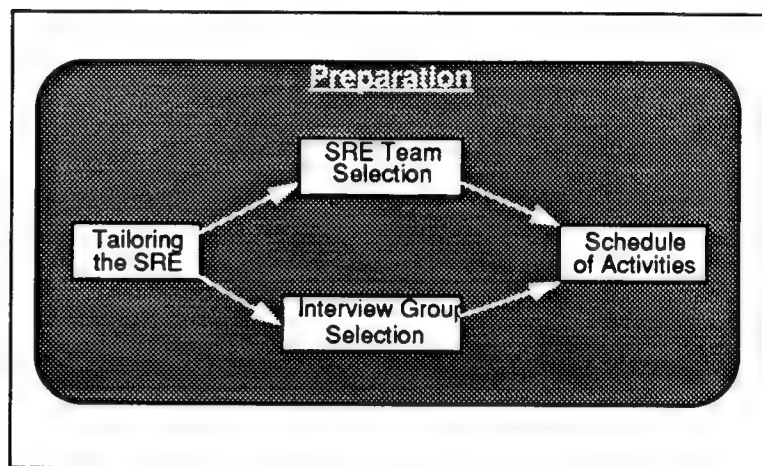
Figure 5-3: Preparation Activities

5.2.1 Tailoring the SRE

Purpose

The Software Risk Evaluation is tailored to the client program or project profile. Tailoring ensures that the SRE will meet the client program or project goals for performing the risk evaluation.

The tailoring activity also ensures that the implementation details of the SRE method can be effectively applied to the program or project specific environment and life-cycle stage.

Specifics

The tailoring activity makes use of the map of the client's goals to the risk taxonomy that was developed earlier during the commitment phase.

The tailoring activity includes identifying the type and number of groups that should be interviewed for the SRE, and tailoring the Taxonomy-Based Questionnaire to meet the goals of the client program or project.

5.2.2 SRE Team Selection

Team Composition

The Software Risk Evaluation team is composed of three or four individuals from an independent organization and two or three individuals from the client organization who are not directly associated with the target program or project.

The client members of the team should not have a reporting relationship with the people who are being interviewed. The client members should not be involved with routine activities of the program or project but should have a knowledge of its organization and technical and business aspects.

Team Skills

SRE team members should possess good communication and interpersonal skills.

Jointly, the team must have a working knowledge in the areas of software development processes, the technology used on the project, the application domain, and relevant organizational procedures.

The team should have undergone a formal training course prior to executing the Software Risk Evaluation.

Team Model

Ideally, the SRE team should function as a structured open team as described in Constantine [Constantine 93]. This involves both collaborative team work and consensus engineering.

Some of the special features in the team model are:

• A catalog of essential team roles is identified.
• Formal specification of and institutionalization of functional roles is given.
• Rotation of roles is practiced to promote flexibility and skill acquisition.
• Default assignment of roles should be made to assure essential functions are performed.
• A structured and organized record of the group decisions and rationale is necessary.
• Technical consensus building is more important than majority-oriented decision making.

Team Selection

The SRE team leader is responsible for forming the team and for assigning specific roles to individuals during the execution phase.

5.2.3 Interview Group Selection

Interview Groups

The SRE team conducts interviews of many different groups associated with the program or project. The interview groups may be formed by organization, region, products, customers and contractors, or software development and acquisition functions.

Functional Groups

The following are some of the typical functional groups that may be represented during a risk evaluation:

- Software designers, developers, and testers
- Functional support personnel such as independent verification & validation (IV&V), quality assurance (QA), configuration management (CM)
- Technical leaders
- Systems and software engineers
- Customers and users
- Contractors
- Project managers, functional support managers, and program managers

Selection Guidelines

The individuals for the interview groups are selected based on both their knowledge of the function they perform and the environment of the target program or project.

Ideally, individuals who are selected are willing and able to express themselves in a group setting. Sometimes, however, they are not. It is the responsibility of the SRE team to ensure the responses.

The individuals within each interview group should not have any direct reporting relationships among themselves. This is to ensure that they are in an environment where they can express their risks, issues, and concerns without any fear of attribution.

Group Size

The interview groups usually consist of five or six individuals for various reasons such as to:

- manage the dynamics of a group interview by the interviewer
- provide sufficient time for follow-up questions
- ensure adequate coverage of the Taxonomy-Based Questionnaire
- effectively observe and engage all respondents by the SRE team

Selection

The site coordinator working under the guidance of the SRE team leader is responsible for identifying and scheduling individuals for each interview session.

The site coordinator and SRE team leader are also responsible for ensuring that interview groups are selected using the guidelines of the SRE method.

5.2.4 Schedule of Activities

Introduction

The schedule of activities for the SRE team's on-site activities is planned sufficiently in advance so that individuals can attend the sessions.

Although some of the sessions must be completed before others can begin, the schedule can be adapted to meet the needs of the program or project. The site coordinator working jointly with the SRE team leader establishes the schedule. The site coordinator also ensures that rooms for the sessions are scheduled and communicated to the participants.

Typical Schedule   Figure 5-4 shows an example of a typical SRE schedule of activities.
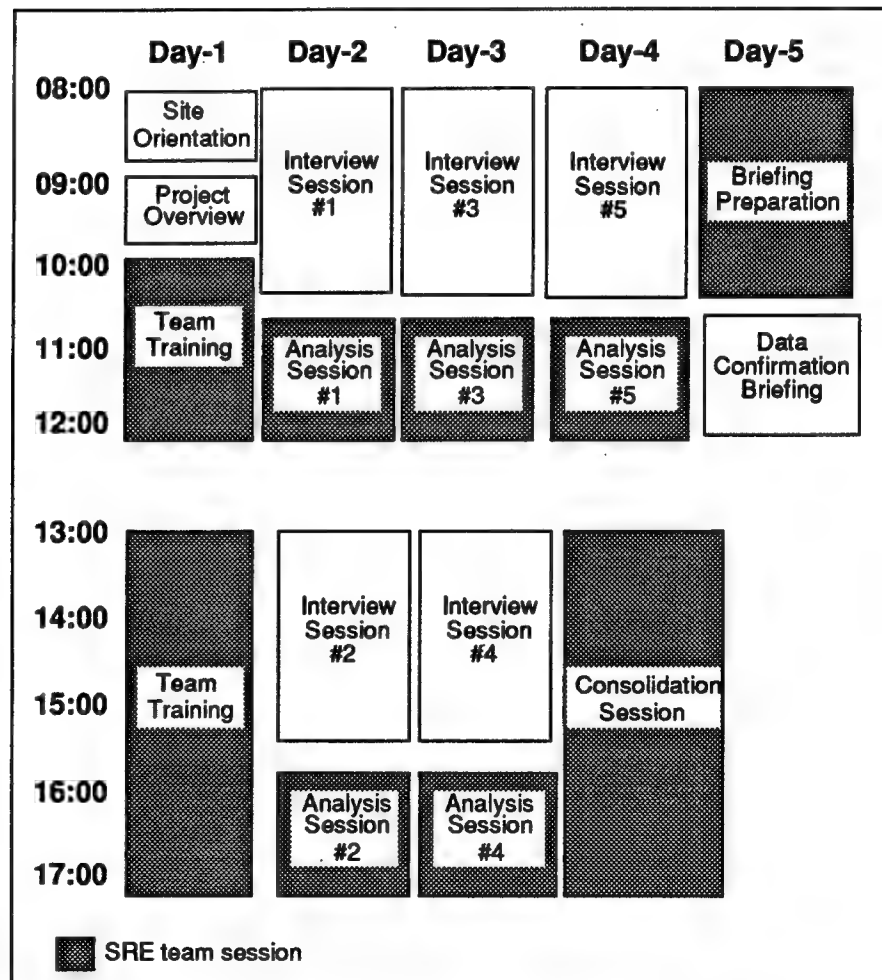
Figure 5-4: Typical Schedule of Activities

5.3 Execution

Introduction

The execution phase consists of those activities that are performed during a site visit of the SRE team to the location of the target program or project.

Figure 5-5 shows the sequence of activities in this phase.
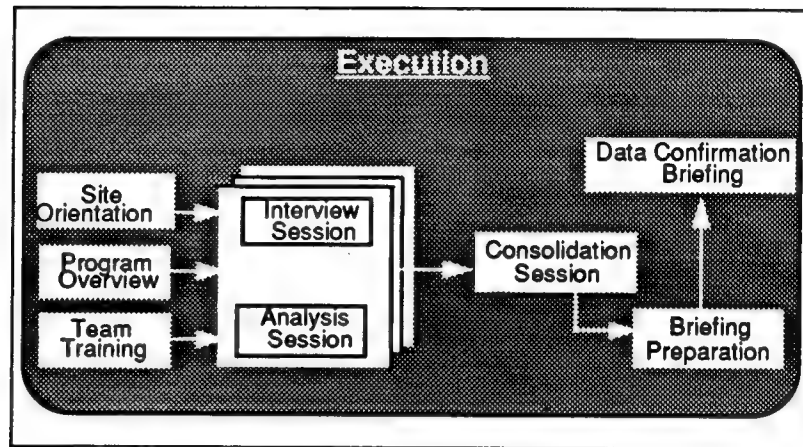
Figure 5-5: Execution Execution Activities

5.3.1 Site Orientation

Purpose

The purpose of the site orientation is for all individuals participating in the execution phase activities to be aware of:

- the objectives for performing the evaluation
- the implementation phases of the SRE
- the individual's role in the implementation phases

Participants

This session is for all participants who are involved in the execution phase activities including management and technical personnel associated with the program or project and the SRE team.

The SRE team leader is responsible for conducting the site orientation.

5.3.2 Program Overview Session

Purpose

The purpose of the overview session is for knowledgeable representatives to present in summary the organization's structure, context, and the technical aspects of the target program or project.

The objective is for the SRE team to review the organizational structure and the characteristics, terminology, and functional aspects of the target software development project.

Participants

This session is open to all participants involved in the risk evaluation including management and technical personnel associated with the target project.

5.3.3 SRE Team Training

Purpose

The purpose of the SRE team training is for all members to have a practical knowledge of the SRE method prior to its execution.

Training areas include the SEI risk paradigm, taxonomy of software development risks, and the Taxonomy-Based Questionnaire.

Techniques

The training also provides skills required to perform the risk evaluation functions and its implementation specifics.

For example, the training prepares the team in the use of interviewing techniques and risk identification skills using simulated scenarios and role playing exercises. The exercises give team members the practice required for interviewing and exposes them to some of the typical behaviors that can be expected during the actual sessions.

5.3.4 Interview Sessions

Purpose

The purpose of the interview sessions is to perform risk detection and specification.

Interview

Each interview session begins with the interviewer using an introductory script followed by questions from the Taxonomy-Based Questionnaire (TBQ).

The process is designed to facilitate the detection of risks on the basis of the participants' discussion rather than by rigidly following the structure of the TBQ. The participants are encouraged to follow any thread of discussion or thought as long as they are objectively discussing potential risks when responding to the questions, or to subsequent follow-up probes, or to cues, or begin to discuss among themselves.

Risk Recording

When a risk is identified the risk recorder will document the risk.

The risk recorder will use the wording of the respondents, and if the meaning is not clear will clarify the statement before documenting it.

The risk recorder or another designated person is responsible for entering the risks into an automated analysis tool.

Session Notes

A session recorder takes notes on the context and other pertinent discussions during the risk detection activities.

Other SRE team members also take notes to ensure the following:

- Any potential risk, issue, or concern that was raised by an interviewee is not overlooked.

- The source of the risk is clearly identifiable and can be tagged to a category in the SEI taxonomy of software development risks.
- Sufficient information is available for the team to make an objective assessment of each risk that was detected.

5.3.5 Analysis Sessions

Purpose

The purpose of the analysis session is to complete the functions of risk specification and risk assessment.

The analysis session is performed by the SRE team. This session is performed immediately after each interview session when the context of the risk is still fresh in the minds of the SRE team members.

Analysis Tasks

Each team member is provided a copy of the recorded risks. The SRE team discusses only those risks where the wording of the risk statements is of concern and may need to be changed.

The team members individually score each risk using the selected assessment mechanism. The team then discusses those risks that have a significant deviation in their scores and reaches consensus.

During the analysis session each risk will also be tagged to a taxonomy group. That is, it will be tagged as belonging to a specific class-element-attribute in the software risk taxonomy.

Analysis Recording   During this session the risk recorder or a designated person enters the risk statement corrections, risk assessment scores, and source of risk categories into the automated analysis tool.

A session recorder documents the team's decisions and rationale throughout the analysis session.

5.3.6 Consolidation Session

Purpose   The purpose of this session is to perform the risk consolidation and if necessary, revise the assessments of the consolidated risks.

This session is performed by the SRE team and is held after all the analysis sessions have been completed.

Consolidation Tasks   Each SRE team member is provided with a copy of the risks from the analysis sessions sorted by their levels of magnitude within each risk category.

The SRE team jointly examine the risks within each category to determine if there are candidates for consolidation. The team reaches consensus on the wording and revisits the assessment scores of the consolidated risks if necessary.

Recording

A risk recorder enters the consolidated risk statements and their revised risk assessments into the automated analysis tool.

A session recorder is responsible for taking notes of the session and documenting the team's decisions and rationale for them.

5.3.7 Briefing Preparation

Purpose

The purpose of this session is to prepare for the data confirmation briefing to be presented at the site.

Contents

The contents of the briefing are a listing of the risk statements that were identified during the execution phase and sorted by their source of risk categories and levels of magnitude.

Although all risk findings may be presented, the focus of the data confirmation briefing should be on the more important ones, that is, on those risks that were assessed at a high level of magnitude.

Review

The briefing preparation includes a review of the contents of the slides that will be presented. This is done to ensure accuracy and verify non-attribution and confidentiality of the source of data.

The person responsible for the briefing should make the necessary corrections to the slides.

5.3.8 Data Confirmation Briefing

Purpose

The purpose of the data confirmation briefing is to validate the risk findings with the organization's management and all individuals who participated in the execution phase activities.

All individuals who were involved in the execution phase activities including the management and technical personnel associated with the target program or project and the SRE team members are encouraged to attend the briefing session.

The briefing session provides feedback to the organization by openly communicating the risks that were found and provides an opportunity for the data gathered at the site to be validated.

Briefing Slides

After the briefing a copy of the slides that were presented is provided to the target organization's management.

Any errors on the slides are communicated to the team leader who ensures the corrections are made in the SRE records.

5.4 Mitigation

Introduction

The mitigation phase consists of activities that are performed to develop the final report for the client.

The mitigation phase activities are performed jointly by the SRE team and a representative group of individuals from the target program or project.

Typically one or two site visits are necessary to perform these activities.

Figure 5-6 shows the sequence of activities in this phase.
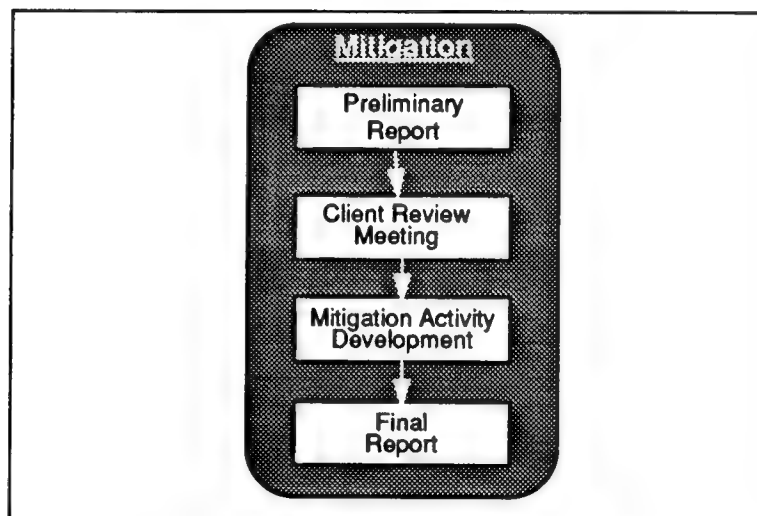
Figure 5-6: Mitigation Phase Activities

5.4.1 Preliminary Report

Purpose

The purpose of the preliminary report is to present the SRE team's analysis of risk findings and proposed strategies for mitigating the risks.

The preparation of the preliminary report is performed by the SRE team after the execution phase activities are completed.

If there are multiple evaluations or execution phases for the program or project, then an additional consolidation session is performed prior to developing the preliminary report.

Contents

The preliminary report contains the consolidated risk findings, mitigation areas, analysis of risk findings, goals, and proposed mitigation strategies or general recommendations.

Activities

The SRE team works with representatives from the project or program for developing the preliminary report.

The group analyzes the consolidated risk findings and identifies the risk mitigation areas for the program or project. The group also produces a map of the risk findings for their respective mitigation areas and the client's goals. Proposed strategies or general recommendations for mitigating the risks may also be included in the preliminary report.

5.4.2 Client Review Meeting

Purpose

The purpose of this meeting is to review the preliminary report with the client's management and other designees from the program or project.

It is important that the client has had sufficient time to review the preliminary report and is prepared to discuss it at this meeting.

Table 5-3 is a set of suggested topics and outcome for the client review meeting.

Table 5.3: Focus of Client Review Meeting

Topic	Outcome
Introductions	• Participants are introduced. • Participants know what the agenda for the meeting is. • Client is prepared to discuss the preliminary report.
Process Overview	• Client has good understanding of the mitigation process. • Client understands the preliminary report is presented for review and discussion purposes.
Discussion: Client Context	Preliminary report is discussed in terms of: • Does it address the issues/risks facing the organization? • Does it address the objectives for successful completion of the program or project? • Does it meet the expectations/goals that were set at the beginning of the risk evaluation?

Table 5.3: Focus of Client Review Meeting (continued)

Topic	Outcome
Discussion: Mitigation Areas/ Recommendations	Preliminary report is discussed in terms of the mitigation areas that were identified and proposed or general recommendations if any: • Are the mitigation areas that were identified linked to the goals and objectives of the organization? • Are the recommendations practical in the context of the client's goals and objectives? • Are there any issues that are outstanding or have not been addressed?
Discussion: Priorities	• What priorities would the client assign to the recommendations? What are the resources that are available? What are the time frames? • What kind of actions is management prepared to take?
Summary	• Summarize the discussions and agreements. • Outline the process for mitigation activity development. • Identify date, time, and people for the next step.

5.4.3 Mitigation Activity Development

Purpose

The purpose of the mitigation activity development is to facilitate the process of developing a risk mitigation plan for the program or project. The process includes identifying recommendations and activities as well as planning details for mitigating the risks.

Mitigation Plan

The mitigation activity development should be of sufficient detail to form the basis for implementing the mitigation activities within the program or project. In this regard it will identify the sequence of activities, costs and benefits, and performance measures. The client management can then decide who should perform the activities, what resources should be allocated, and when each activity should be completed.

Participants

The mitigation activity development requires program- or project-specific knowledge that is necessary to ensure the mitigation activities are practical.

In addition to the program or project representatives who participated in producing the preliminary report, other individuals who can provide the required technical or business knowledge may also be required to participate.

Logistics

The mitigation activity development may be completed in a single 2- or 3-day working session, or spread out over many meetings of shorter durations.

Although the meetings can be held at any location, usually the SRE team and other representatives visit the site of the program or project. This is not only convenient for referencing materials from the program or project but also for inviting other knowledgeable members from the site, if the need arises.

Process

For each mitigation area the group will discuss and identify the specific recommendation and the primary activities that are required to be performed. The group will also identify constraints within the program or project environment that are barriers for implementing the primary activities.

When there are constraints the group will "brainstorm" to identify those activities that should be performed to overcome the barriers. The primary activities and those that are required to overcome any constraints are termed *key activities* in the SRE method.

In addition the group will identify other activities that are required for effectively implementing the recommendation. Some examples of the other activities are producing task plans or conducting kick-off meetings. These activities are termed *enabling activities.*

The process of mitigation activity development also includes sequencing the activities, developing costs and benefits, identifying performance measures, and other details necessary to facilitate the mitigation plan for the program or project.

5.4.4 Final Report

Purpose

The purpose of the final report is to provide the results of the risk evaluation to the management of the client program or project.

Contents

The final report contains an executive summary, the general recommendations or strategies for risk mitigation, and the specific mitigation activities that are necessary to address the risks that were identified for the program or project.

The contents of the final report will facilitate the completion of the mitigation plan that can be used for mitigating the risks of the program or project.

The final report includes other topics that were provided in the preliminary report with the modifications based on client review. These are risk findings, mitigation areas and analysis of findings, mapping of risks to mitigation areas, and target goals for each mitigation area.

A copy of all briefing slides and other materials that were used in the risk evaluation are also provided as an appendix to the final report.

6   Automated Tools

Introduction

The use of automated tools to support the risk evaluation activities is recommended as it ensures that the functions will be performed efficiently.

Some basic tool support is essential for completing the tasks within the limited resources and time that are available to the SRE team during the execution phase.

Analysis Support

In the example shown in Figure 6-1, risk statements that are identified after each interview session are entered into the basic tool. This exercise is done prior to the beginning of each analysis session. The basic tool is then used to print the analysis sheets that will be used during the analysis session.

Later, after the analysis, the results of the session are entered into the tool and it is used to calculate the aggregate scores and level of magnitude of the risks.

Consolidation Support

As shown in Figure 6-1, before the consolidation session the tool is used to print a complete listing of the risks, sorted by their source of risk and level of magnitude.

This listing is used by the SRE team during the consolidation session.

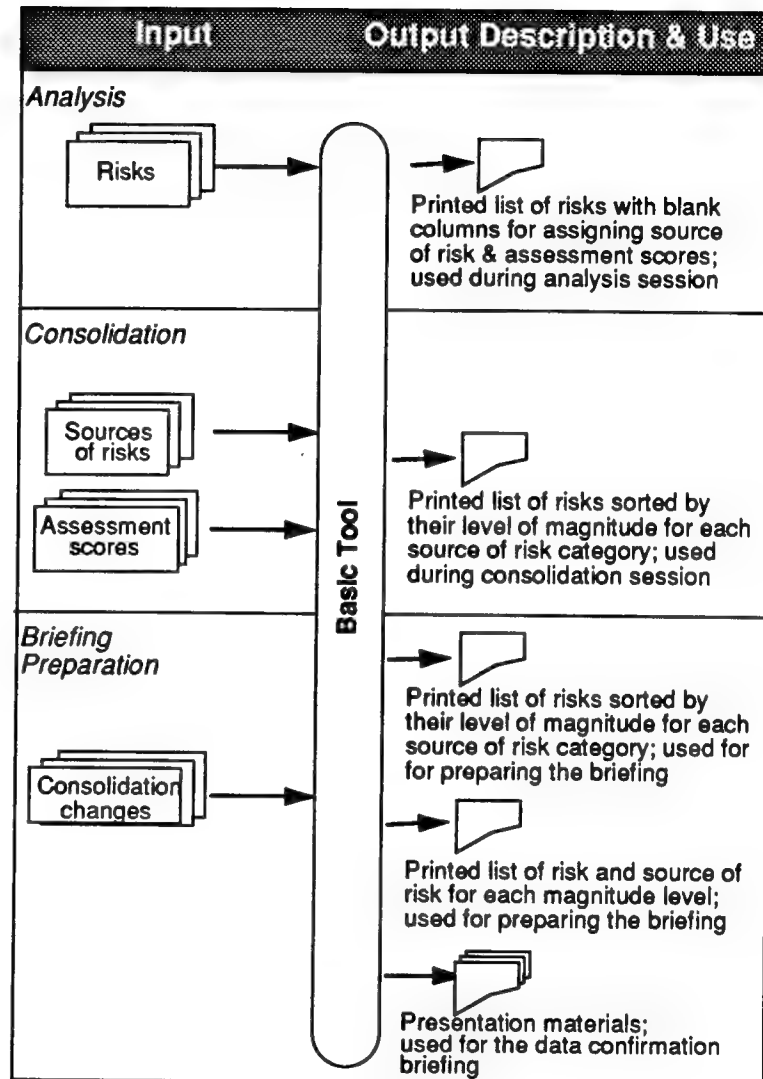| Input	Output Description & Use
Analysis	
Risks	Printed list of risks with blank columns for assigning source of risk & assessment scores; used during analysis session
Consolidation	
Sources of risks	
Assessment scores	Printed list of risks sorted by their level of magnitude for each source of risk category; used during consolidation session
Briefing Preparation	Printed list of risks sorted by their level of magnitude for each source of risk category; used for for preparing the briefing
Consolidation changes	Printed list of risk and source of risk for each magnitude level; used for preparing the briefing
	Presentation materials; used for the data confirmation briefing

Figure 6-1   Use of Basic Tools

Briefing Support

After the consolidation session any changes are entered into the basic tool. The tool is then used to print out reports and other presentation materials that are necessary for the data confirmation briefing.

7 Post-Evaluation Activities

7.1 Improvement and Enhancement

Improvement Opportunities

After each risk evaluation, the team meets to discuss its experience and the lessons learned during implementation. The purpose of this exercise is to identify improvement opportunities for the SRE method. Modifications to the existing activities and any new requirements are processed through a formal change control mechanism for implementation. This ensures continuous improvement and enhancement to the SRE method.

7.2 Knowledge Engineering

Historical Data

The SRE results are normalized and stored in a database containing historical risk data. This is performed by ensuring the data will maintain the confidentiality of the program or project. Any proprietary trade information of the client organization will not be stored in the database. The data are also rendered non-attributing, for example, by removing any project-specific references in the risk statements.

Domain Data

Domains such as characteristics of the project or program, the structure and interactions among the technical risks that were identified, cognitive and intellectual processes must be documented and analyzed as well.

References

[Airforce 88]   Air Force Systems Command and Air Force Logistics Command. *Acquisition Management Software Risk Abatement*. AFSC/AFLC Pamphlet 800-45, September 30, 1988.

[Boehm 81]   Boehm, Barry W. "Statistical Decision Theory: The Value of Information," *Software Engineering Economics* , Englewood Cliffs, NJ.: Prentice-Hall Inc., 1981.

[Boehm 89]   Boehm, Barry W. *Tutorial: Software Risk Management*. IEEE, Washington DC: Computer Society Press, 1989.

[Boehm 91]   Boehm Barry W. "Software risk Management: Principles and Practices," *IEEE Software*, (January 1991): 32-41.

[Carr 93]   Carr M.; Konda S.; Monarch I.; Ulrich C.; & Walker C. *Taxonomy-Based Risk Identification* (CMU/SEI-93-TR-6, ADA266992). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1993.

[Constantine 93]   Constantine, Larry L. "Work Organization: Paradigm for Project Management and Organization," *Communications of the ACM 6*, 10. (October 1993): 34-43.

[Dym 91]   Dym, C. L.; & Levitt R. E., *Knowledge-Based Systems in Engineering*. New York: McGraw-Hill, 1991.

[Gluch 93]	Gluch, David. *A Construct for Managing Software Development Risks.* Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, draft technical report, October 27, 1993.
[Morgan 81]	Morgan, Granger M. "Probing the Question of Technology-Induced Risk". *IEEE Spectrum 18*, 11. (1981): 58-64.
[Mulvehill 88]	Mulvehill A. M. *The Evolution of a Knowledge-Base.* Bedford, Ma.: The Mitre Corporation, Dept. D78 Project No. 5450, NASA Contract No. F19628-89-C-0001, October 1988.
[Paulk 93]	Paulk, M.C.; Curtis, B.; Chrissis, M.B.; & Weber, C.V. *Capability Maturity Model For Software,Version 1.1.* Pittsburgh, PA.: Software Engineering Institute, Carnegie Mellon University, (CMU/SEI-93-24), February 1993.
[Poltrock 89]	Poltrock, Steven E.; & Nasr, Maria G. *Protocol Analysis: A Tool for Analyzing Human-Computer Interactions.* Austin, Tx.: Microelectronics and Computer Technology Corporation, (MCC Technical Report Number ACT-H1-186-89), May 1989.

Appendix A:
Taxonomy of Software Risk

This appendix provides the definitions of the Software Risk Taxonomy's class, element, and attributes.

Figure A-1 shows an overview of the taxonomy of software risk.

A   Product Engineering

Product engineering refers to the system engineering and software engineering activities involved in creating a system which satisfies specified requirements and, also, customer expectations. These activities include system and software requirements analysis and specification, software design and implementation, integration of hardware and software components, and software and system test.

The elements of this class cover traditional software engineering activities. They comprise those technical factors associated with the deliverable product itself, independent of the processes or tools used to produce it or the constraints imposed by finite resources or external factors beyond program control.

Product engineering risks generally result from requirements which are technically difficult or impossible to implement, often in combination with inability to negotiate relaxed requirements or revised budgets and schedules; from inadequate analysis of requirements or design specifications, or from poor quality design or coding specifications.

1   Requirements

Attributes of the requirements element cover both the quality of the requirements specification and, also, the difficulty of implementing a system which satisfies the requirements.

A. Product Engineering
  1. Requirements
     a. Stability
     b. Completeness
     c. Clarity
     d. Validity
     e. Feasibility
     f. Precedent
     g. Scale
  2. Design
     a. Functionality
     b. Difficulty
     c. Interfaces
     d. Performance
     e. Testability
     f. Hardware Constraints
     g. Non-Developmental Software
  3. Code and Unit Test
     a. Feasibility
     b. Testing
     c. Coding/Implementation
  4. Integration and Test
     a. Environment
     b. Product Integration
     c. System Integration
  5. Engineering Specialties
     a. Maintainability
     b. Reliability
     c. Safety
     d. Security
     e. Human Factors
     f. Specifications

B. Development Environment
  1. Development Process
     a. Formality
     b. Suitability
     c. Process Control
     d. Familiarity
     e. Product Control
  2. Development System
     a. Capacity
     b. Suitability
     c. Usability
     d. Familiarity
     e. Reliability
     f. System Support
     g. Deliverability
  3. Management Process
     a. Planning
     b. Project Organization
     c. Management Experience
     d. Program Interfaces
  4. Management Methods
     a. Monitoring
     b. Personnel Management
     c. Quality Assurance
     d. Configuration Management
  5. Work Environment
     a. Quality Attitude
     b. Cooperation
     c. Communication
     d. Morale

C. Program Constraints
  1. Resources
     a. Schedule
     b. Staff
     c. Budget
     d. Facilities
  2. Contract
     a. Type of contract
     b. Restrictions
     c. Dependencies
  3. Program Interfaces
     a. Customer
     b. Associate Contractors
     c. Subcontractors
     d. Prime Contractor
     e. Corporate Management
     f. Vendors
     g. Politics

Figure A-1 Taxonomy of Software Risk - An Overview

The following attributes characterize the requirements element.

a) Stability

The stability attribute refers to the degree to which the requirements are changing and the possible effect changing requirements and external interfaces will have on the quality, functionality, schedule, design, integration, and testing of the product being built.

The attribute also includes issues which arise from the inability to control rapidly changing requirements. For example, impact analyses may be inaccurate because it is impossible to define the baseline against which the changes will be implemented.

b) Completeness

Missing or incompletely specified requirements may appear in many forms, such as: a requirements document with many functions or parameters "to be defined," requirements that are not specified adequately to develop acceptance criteria, or inadvertently omitted requirements. When missing information is not supplied in a timely manner, implementation may be based on contractor assumptions which differ from customer expectations.

When customer expectations are not documented in the specification, they are not budgeted into the cost and schedule.

c)   Clarity

This attribute refers to ambiguously or imprecisely written individual requirements which are not resolved until late in the development phase. This lack of a mutual contractor and customer understanding may require re-work to meet the customer intent for a requirement.

d)   Validity

This attribute refers to whether the aggregate requirements reflect customer intentions for the product. This may be affected by misunderstandings of the written requirements by the contractor or customer, unwritten customer expectations or requirements, or a specification in which the end user did not have inputs.

This attribute is affected by the completeness and clarity attributes of the requirements specifications, but refers to the larger question of the system as a whole meeting customer intent.

e) Feasibility

The feasibility attribute refers to the difficulty of implementing a single technical or operational requirement, or of simultaneously meeting conflicting requirements. Sometimes two requirements are by themselves are feasible, but together are not; they cannot both exist in the same product at the same time.

Also included is the ability to determine an adequate qualification method for demonstration that the system satisfies the requirement.

f) Precedent

The precedent attribute concerns capabilities that have not been successfully implemented in any existing systems or are beyond the experience program personnel or of the company. The degree of risk depends on allocation of additional schedule and budget to determine the feasibility of their implementation; contingency plans in case the requirements are not feasible as stated; and flexibility in the contract to allocate implementation budget and schedule based on the outcome of the feasibility study.

Even when unprecedented requirements are feasible, there may still be a risk of underestimating the difficulty of implementation and committing to an inadequate budget and schedule.

g)   Scale

This attribute covers both technical and management challenges presented by large complex systems development.

Technical challenges include satisfaction of timing, scheduling and response requirements, communication among processors, complexity of system integration, analysis of inter-component dependencies, and impacts due to changes in requirements.

Management of a large number of tasks and people introduces a complexity in such areas as project organization, delegation of responsibilities, communication among management and peers, and configuration management.

2   Design

The attributes of the design element cover the design and feasibility of algorithms, functions or performance requirements and of the internal and external product interfaces. Difficulty in testing may begin here with failure to work to testable requirements or to include test features in the design.

The following attributes characterize the design element.

a) Functionality

This attribute covers functional requirements which may not submit to a feasible design, or use of specified algorithms or designs without a high degree of certainty that they will satisfy their source requirements. Algorithm and design studies may not have used appropriate investigation techniques or may show marginal feasibility.

b) Difficulty

The difficulty attribute refers to functional or design requirements that may be extremely difficult to realize. Systems engineering may design a system architecture difficult to implement, or requirements analysis may have been based on optimistic design assumptions.

The difficulty attribute differs from design feasibility in that it does not proceed from pre-ordained algorithms or designs.

c) Interfaces

This attribute covers all hardware and software interfaces that are within the scope of the development program including interfaces between configuration items, and the existence of techniques for defining and managing the interfaces.

Special note is taken of non-developmental software and developmental hardware interfaces.

d) Performance

The performance attribute refers to all aspects of performance: user and real-time response requirements, throughput requirements, performance analyses, and performance modeling throughout the development cycle.

e) Testability

The testability attribute covers the amenability of the design to testing, design of features to facilitate testing, and the inclusion in the design process of people who will design and conduct product tests.

f) Hardware Constraints

This attribute covers target hardware with respect to system and processor architecture, and the dependence on hardware to meet system and software performance requirements. These constraints may include throughput or memory speeds, real-time

response capability, database access or capacity limitations, insufficient reliability, unsuitability to system function, or insufficiency in the amount of specified hardware.

g) Non-Developmental Software

Since non-developmental software (NDS) is not designed to system requirements, but selected as a "best fit," it may not conform precisely to performance, operability or supportability requirements.

The customer may not accept vendor or developer test and reliability data to demonstrate satisfaction of the requirements allocated to NDS. It may then be difficult to produce this data to satisfy acceptance criteria and only within the estimated NDS test budget.

Requirements changes may necessitate reengineering or reliance on vendors for special purpose upgrades.

3 Code and Unit Test

Attributes of this element are associated with the quality and stability of software or interface specifications, and the constraints which may present implementation or test difficulties.

a)   Feasibility

The feasibility attribute of the code and unit test element addresses possible difficulties which may arise from poor design or design specification or from inherently difficult implementation needs.

For example, the design may not have quality attributes such as module cohesiveness or interface minimization; the size of the modules may contribute to design complexity; the design may not be specified in sufficient detail, requiring the programmer to make assumptions or design decisions during coding; or the design and interface specifications may be changing, perhaps without an approved detailed design baseline; and, the use of developmental hardware may make an additional contribution to inadequate or unstable interface specification. Or, the nature of the system itself may aggravate the difficulty and complexity of the coding task.

b)   Unit Test

Factors affecting unit test include planning and preparation and also the resources and time allocated for test.

Constituents of these factors are: entering unit test with quality code obtained from formal or informal code inspection or verification procedures; pre-planned test cases which have been verified to test unit

requirements; a test bed consisting of the necessary hardware or emulators, and software or simulators, and test data to satisfy the planned test; sufficient schedule to plan and carry out the test plan.

c) Coding/Implementation

This attribute addresses the implications of implementation constraints. Some of these are: target hardware which is marginal or inadequate with regard to speed, architecture, memory size or external storage capacity; required implementation languages or methods; or differences between the development and target hardware.

4 Integration and Test

This element covers integration and test planning, execution and facilities for both the contractual product and for the integration of the product into the system or site environment.

a) Environment

The integration and test environment includes the hardware and software support facilities and adequate

test cases reflecting realistic operational scenarios, and realistic test data and conditions.

This attribute addresses the adequacy of this environment to enable integration in a realistic environment or to fully test all functional and performance requirements.

b)   Product Integration

The product integration attribute refers to integration of the software components to each other and to the target hardware, and testing of the contractually deliverable product. Factors which may affect this are internal interface specifications for either hardware or software, testability of requirements, negotiation of customer agreement on test criteria, adequacy of test specifications, and sufficiency of time for integration and test.

c)   System Integration

The system integration attribute refers to integration of the contractual product to interfacing systems or sites. Factors associated with this attribute are external interface specifications, ability to faithfully produce system interface conditions prior to site or system integration, access to the system or site being interfaced to, adequacy of time for testing, and associate contractor relationships.

5   Engineering Specialities

The engineering specialty requirements are treated separately from the general requirements element primarily because they are often addressed by specialists who may not be full-time on the program. This taxonomic separation is a device to ensure that these specialists are called in to analyze the risks associated with their areas of expertise.

a)   Maintainability

Maintainability may be impaired by poor software architecture, design, code, or documentation resulting from undefined or un-enforced standards and requirements, or from neglecting to analyze the system from a maintenance point of view.

b)   Reliability

System reliability or availability requirements may be affected by hardware not meeting its reliability specifications or system complexity which aggravates difficulties in meeting recovery timeliness. Reliability or availability requirements allocated to software may be stated in absolute terms, rather than as separable from hardware and independently testable.

c) Safety

This attribute addresses the difficulty of implementing allocated safety requirements and also the potential difficulty of demonstrating satisfaction of requirements by faithful simulation of the unsafe conditions and corrective actions. Full demonstration may not be possible until the system is installed and operational.

d) Security

This attribute addresses lack of experience in implementing the required level of system security which may result in under-estimation of the effort required for rigorous verification methods, certification and accreditation, and secure or trusted development process logistics; developing to unprecedented requirements; and dependencies on delivery of certified hardware or software.

e) Human Factors

Meeting human factors requirements is dependent on understanding the operational environment of the installed system and agreement with various customer and user factions on a mutual understanding of the expectations embodied in the human factors requirements.

It is difficult to convey this understanding in a written specification. Mutual agreement on the human interface may require continuous prototyping and demonstration to various customer factions.

f) Specifications

This attribute addresses specifications for the system, hardware, software, interface, or test requirements or design at any level with respect to feasibility of implementation and the quality attributes of stability, completeness, clarity and verifiability.

B   Development Environment

The development environment class of the Software Development Risk Taxonomy addresses the project environment and the process used to engineer a software product. This environment includes the development process and system, management methods, and work environment. These environmental elements are characterized below by their component attributes.

1   Development Process

The development process element refers to the process by which the contractor proposes to satisfy the customer's requirements. The process is the sequence of steps—the inputs, outputs, actions, validation criteria, and monitoring activities— leading from the initial requirement specification to the final delivered product. The development process includes such phases as requirements analysis, product definition, product creation, testing, and delivery. It includes both general management processes such as costing, schedule tracking, and personnel assignment, and also project-specific processes such as feasibility studies, design reviews, and regression testing.

This element groups risks that result from a development process that is inadequately planned, defined and documented; that is not suited to the activities necessary to accomplish the project goals; and that is poorly communicated to the staff and lacks enforced usage.

a) Formality

Formality of the development process is a function of the degree to which a consistent process is defined, documented, and communicated for all aspects and phases of the development.

b) Suitability

Suitability refers to the adequacy with which the selected development model, process, methods and tools support the scope and type of activities required for the specific program.

c) Process Control

Process control refers not only to ensuring usage of the defined process by program personnel, but also to the measurement and improvement of the process based on observation with respect to quality and productivity goals. Control may be complicated due to distributed development sites.

d) Familiarity

Familiarity with the development process covers knowledge of, experience in, and comfort with the prescribed process.

e) Product Control

Product control is dependent on traceability of requirements from the source specification through implementation, such that the product test will demonstrate the source requirements. The change control process makes use of the traceability mechanism in impact analyses and reflects all resultant document modifications including interface and test documentation.

2 Development System

The development system element addresses the hardware and software tools and supporting equipment used in product development. This includes CASE tools, simulators, compilers, test equipment, and host computer systems.

a) Capacity

Risks associated with the capacity of the development system may result from too few workstations, insufficient processing power or database storage, or other inadequacies in equipment to support parallel activities for development, test and support activities.

b) Suitability

Suitability of the development system is associated with the degree to which it is supportive of the specific development models, processes, methods, procedures and activities required and selected for the program. This includes the development, management, documentation, and configuration management processes.

c) Usability

Usability refers to development system documentation, accessibility and workspace as well as ease of use.

d)   Familiarity

Development system familiarity depends on prior use of the system by the company and by project personnel as well as adequate training for new users.

e)   Reliability

Development system reliability is a measure of whether the needed components of the development system are available and working properly whenever required by any program personnel.

f)   System Support

Development system support involves training in use of the system, access to expert users or consultants, and repair or resolution of problems by vendors.

g)   Deliverability

Some contracts require delivery of the development system. Risks may result from neglecting to bid and allocate resources to ensure that the development system meets all deliverable requirements.

3   Management Process

The management process element pertains to risks associated with planning, monitoring and controlling budget and schedule; with controlling factors involved in definition, implementation, and test of the product; with management of project personnel; and with handling external organizations including the customer, senior management, matrix management and other contractors.

a)   Planning

The planning attribute addresses risks associated with developing a well-defined plan which is responsive to contingencies as well as long range goals and which was formulated with the input and acquiescence of those affected by it. Also addressed are managing according to the plan and formally modifying the plan when changes are necessary.

b)   Project Organization

This attribute addresses the effectiveness of the program organization, the effective definition of roles and responsibilities, and the assurance that these roles and lines of authority are understood by program personnel.

c) Management Experience

This attribute refers to the experience of all levels of managers with respect to management, software development management, the application domain, the scale and complexity of the system and program, the selected development process, and hands-on development of software.

d) Program Interfaces

This attribute refers to the interactions of managers at all levels with program personnel at all levels, and with external personnel such as the customer, senior management, and peer managers.

4 Management Methods

This element refers to methods for managing both the development of the product and program personnel. These include quality assurance, configuration management, staff development with respect to program needs, and maintaining communication about program status and needs.

a) Monitoring

The monitoring includes the activities of obtaining and acting upon status reports, allocating status information to the appropriate program organizations, and maintaining and using progress metrics.

b) Personnel Management

Personnel management refers to selection and training of program members and ensuring that they: take part in planning and customer interaction for their areas of responsibility; work according to plan; receive the help they need or ask for to carry out their responsibilities.

c) Quality Assurance

The quality assurance attribute refers to the procedures instituted for ensuring that contractual processes and standards are implemented properly for all program activities, and that the quality assurance function is adequately staffed to perform their duties.

d)   Configuration Management

The configuration management (CM) attribute addresses both staffing and tools for the CM function as well as the complexity of the required CM process with respect to such factors as multiple development and installation sites and product coordination with existing, possibly changing, systems.

5   Work Environment

The work environment element refers to subjective aspects of the environment such as the amount of care given to ensuring that people are kept informed of program goals and information, the way people work together, responsiveness to staff inputs, and the attitude and morale of the program personnel.

a)   Quality Attitude

This attribute refers to the tendency of program personnel to do quality work in general and to conform to specific quality standards for the program and product.

b) Cooperation

The cooperation attribute addresses lack of team spirit among development staff both within and across work groups and the failure of all management levels to demonstrate that best efforts are being made to remove barriers to efficient accomplishment of work.

c) Communication

Risks which result from poor communication are due to lack of knowledge of the system mission, requirements and design goals and methods, or to lack of information about the importance of program goals to the company or the project.

d) Morale

Risks that result from low morale range across low levels of enthusiasm and thus low performance, productivity or creativity; anger which may result in intentional damage to the project or the product; mass exodus of staff from the project; and a reputation within the company which makes it difficult to recruit.

C   Program Constraints

Program constraints refer to the "externals" of the project. These are factors that may be outside the control of the project but can still have major effects on its success or constitute sources of substantial risk.

1   Resources

This element addresses resources for which the program is dependent on factors outside program control to obtain and maintain. These include schedule, staff, budget, and facilities.

a)   Schedule

This attribute refers to the stability of the schedule with respect to internal and external events or dependencies and the viability of estimates and planning for all phases and aspects of the program.

b)   Staff

This attribute refers to the stability and adequacy of the staff in terms of numbers and skill levels, their experience and skills in the required technical areas

and application domain, as well as their availability when needed.

c)   Budget

This attribute refers to the stability of the budget with respect to internal and external events or dependencies and the viability of estimates and planning for all phases and aspects of the program.

d)   Facilities

This attribute refers to the adequacy of the program facilities for development, integration, and testing of the product.

2   Contract

Risks associated with the program contract are classified according to contract type, restrictions, and dependencies.

a) Type of contract

This attribute covers the payment terms (cost plus award fee, cost plus fixed fee,...) and the contractual requirements associated with such items as the Statement of Work, Contract Data Requirements List, and the amount and conditions of customer involvement.

b) Restrictions

Contract restrictions and restraints refer to contractual directives to, for example, use specific development methods, standards, or equipment and the resultant complications such as acquisition of data rights for use of non-developmental software.

c) Dependencies

This attribute refers to the possible contractual dependencies on outside contractors or vendors, customer furnished equipment or software, or other outside products and services.

3 Program Interfaces

This element consists of the various interfaces with entities and organizations outside the development program itself.

a) Customer

The customer attribute refers to the customer's level of skill and experience in the technical or application domain of the program as well as difficult working relationships or poor mechanisms for attaining customer agreement and approvals, not having access to certain customer factions, or not being able to communicate with the customer in a forthright manner.

b) Associate Contractors

The presence of associate contractors may introduce risks due to conflicting political agenda, problems of interfaces to systems being developed by outside organizations, or lack of cooperation in coordination of schedules and configuration changes.

c) Subcontractors

The presence of subcontractors may introduce risks due to inadequate task definitions and subcontractor

management mechanisms, or to not transferring sub-contractor technology and knowledge to the program or corporation.

d) Prime Contractor

When the program is a subcontract, risks may arise from poorly defined task definitions, complex reporting arrangements, or dependencies on technical or pro-grammatic information.

e) Corporate Management

Risks in the corporate management area include poor communication and direction from senior management as well as non-optimum levels of support.

f) Vendors

Vendor risks may present themselves in the forms of dependencies on deliveries and support for critical sys-tem components.

g) Politics

Political risks may accrue from relationships with the company, customer, associate contractors or subcontractors and may affect technical decisions.company, customer, associate contractors or subcontractors and may affect technical decisions.

Appendix B:
Taxonomy-Based Questionnaire

A. Product Engineering

1. Requirements

a) Stability
[Are requirements changing even as the product is being produced?]

[1]   Are the requirements stable?

    (No)   (1.a) What is the effect on the system?
- Quality
- Functionality
- Schedule
- Integration
- Design
- Testing

[2]   Are the external interfaces changing?

b) Completeness
[Are requirements missing or incompletely specified?]

[3]   Are there any TBDs in the specifications?

[4]   Are there requirements you know should be in the specification but aren't?

    (Yes) (4.a) Will you be able to get these requirements into the system?

[5]   Does the customer have unwritten requirements/expectations?

    (Yes) (5.a) Is there a way to capture these requirements?

[6]   Are the external interfaces completely defined?

c) Clarity
[Are requirements unclear or in need of interpretation?]

[7] Are you able to understand the requirements as written?

    (No)   (7.a) Are the ambiguities being resolved satisfactorily?

    (Yes) (7.b) There are no ambiguities or problems of interpretation?

d) Validity
[Will the requirements lead to the product the customer has in mind?]

[8] Are there any requirements that may not specify what the customer really wants?

    (Yes) (8.a) How are you resolving this?

[9] Do you and the customer understand the same thing by the requirements?

    (Yes) (9.a) Is there a process by which to determine this?

[10] How do you validate the requirements?

- Prototyping
- Analysis
- Simulations

e) Feasibility
[Are requirements infeasible from an analytical point of view?]

[11] Are there any requirements that are technically difficult to implement?

    (Yes) (11.a) What are they?

    (Yes) (11.b) Why are they difficult to implement?

    (No)   (11.c) Were feasibility studies done for these requirements?

(Yes) (11.c.1) How confident are you of the assumptions made in the studies?

f) Precedent

[Do requirements specify something never done before, or that your company has not done before?]

[12] Are there any state-of-the-art requirements?

- Technologies
- Methods
- Languages
- Hardware

(No) (12.a) Are any of these new to you?

(Yes) (12.b) Does the program have sufficient knowledge in these areas?

(No) (12.b.1) Is there a plan for acquiring knowledge in these areas?

g) Scale

[Do requirements specify a product larger, more complex, or requiring a larger organization than in the experience of the company?]

[13] Is the system size and complexity a concern?

(No) (13.a) Have you done something of this size and complexity before?

[14] Does the size require a larger organization than usual for your company?

2. Design

a) Functionality

[Are there any potential problems in meeting functionality requirements?]

[15] Are there any specified algorithms which may not satisfy the requirements?

(No) (15.a) Are any of the algorithms or design marginal with respect to meeting requirements?

[16] How do you determine the feasibility of algorithms and designs?

- Prototyping
- Modeling
- Analysis
- Simulation

b)   Difficulty

[Will the design and/or implementation be difficult to achieve?]

[17] Does any of the design depend on unrealistic or optimistic assumptions?

[18] Are there any requirements or functions which are difficult to design?

(No) (18.a) Do you have solutions for all the requirements?

(Yes) (18.b) What are the requirements?

- Why are they difficult?

c)   Interfaces

[Are the internal interfaces (hardware and software) well defined and controlled?]

[19] Are the internal interfaces well defined?

- Software-to-software
- Software-to-hardware

[20] Is there a process for defining internal interfaces?

(Yes) (20.a) Is there a change control process for internal interfaces?

[21] Is hardware being developed in parallel with software?

(Yes) (21.a) Are the hardware specifications changing?

(Yes) (21.b) Have all the interfaces to software been defined?

(Yes) (21.c) Will there be engineering design models that can be used to test the software?

d) Performance
[Are there stringent response time or throughput requirements?]

[22] Are there any problems with performance?

- Throughput
- Scheduling asynchronous real-time events
- Real-time response
- Recovery timelines
- Response time
- Database response, contention, or access

[23] Has a performance analysis been done?

(Yes) (23.a) What is your level of confidence in the performance analysis?

(Yes) (23.b) Do you have a model to track performance through design and implementation?

e) Testability
[Is the product difficult or impossible to test?]

[24] Is the software going to be easy to test?

[25] Does the design include features to aid testing?

[26] Do the testers get involved in analyzing requirements?

f)   Hardware Constraints
[Are there tight constraints on the target hardware?]

[27]   Does the hardware limit your ability to meet any requirements?

- Architecture
- Memory capacity
- Throughput
- Real-time response
- Response time
- Recovery timelines
- Database performance
- Functionality
- Reliability
- Availability

g)   Non-Developmental-Software
[Are there problems with software used in the program but not developed by the program?]

If re-used or re-engineered software exists

[28]   Are you reusing or reengineering software not developed on the program?

(Yes) (28.a) Do you foresee any problems?

- Documentation
- Performance
- Functionality
- Timely delivery
- Customization

If COTS software is being used

[29] Are there any problems with using COTS (commercial off-the-shelf) software?

- Insufficient documentation to determine interfaces, size, or performance
- Poor performance
- Requires a large share of memory or database storage.
- Difficult to interface with application software
- Not thoroughly tested
- Not bug free
- Not maintained adequately
- Slow vendor response

[30] Do you foresee any problem with integrating COTS software updates or revisions?

3. Code and Unit Test

a) Feasibility
[Is the implementation of the design difficult or impossible?]

[31] Are any parts of the product implementation not completely defined by the design specification?

[32] Are the selected algorithms and designs easy to implement?

b) Testing
[Is the specified level and time for unit testing adequate?]

[33] Do you begin unit testing before you verify code with respect to the design

[34] Has sufficient unit testing been specified?

[35] Is there sufficient time to perform all the unit testing you think should be done?

[36] Will compromises be made regarding unit testing if there are schedule problems?

c) Coding/Implementation
[Are there any problems with coding and implementation?]

[37] Are the design specifications in sufficient detail to write the code?

[38] Is the design changing while coding is being done?

[39] Are there system constraints making the code difficult to write?

- Timing
- Memory
- External storage

[40] Is the language suitable for producing the software on this program?

[41] Are there multiple languages used on the program?

(Yes) (41.a) Is there interface compatibility between the code produced by the different compilers?

[42] Is the development computer the same as the target computer?

(No) (42.a) Are there compiler differences between the two?

If developmental hardware is being used

[43] Are the hardware specifications adequate to code the software?

[44] Are the hardware specifications changing while the code is being written?

4. Integration and Test

a) Environment
[Is the integration and test environment adequate?]

[45] Will there be sufficient hardware to do adequate integration and testing?

[46] Is there any problem with developing realistic scenarios and test data to demonstrate any requirements?

- Specified data traffic
- Real-time response
- Asynchronous event handling
- Multi-user interaction

[47] Are you able to verify performance in your facility?

[48] Does hardware and software instrumentation facilitate testing?

(Yes) (48.a)Is it sufficient for all testing?

b) Product
[Is the interface definition inadequate, facilities inadequate, time insufficient?]

[49] Will the target hardware be available when needed?

[50] Have acceptance criteria been agreed to for all requirements?

(Yes) (50.a) Is there a formal agreement?

[51] Are the external interfaces defined, documented, and baselined?

[52] Are there any requirements that will be difficult to test?

[53] Has sufficient product integration been specified?

[54] Has adequate time been allocated for product integration and test?

If COTS

[55] Will vendor data be accepted in verification of requirements allocated to COTS products?

(Yes) (55.a) Is the contract clear on that?

c) System
[System integration uncoordinated, poor interface definition, or inadequate facilities?]

[56] Has sufficient system integration been specified?

[57] Has adequate time been allocated for system integration and test?

[58] Are all contractors part of the integration team?

[59] Will the product be integrated into an existing system?

(Yes) (59.a) Is there a parallel cutover period with the existing system?

(No) (59.a.1) How will you guarantee the product will work correctly when integrated?

[60]   Will system integration occur on customer site?

5.   Engineering Specialties

a)   Maintainability
[Will the implementation difficult to understand or maintain?]

[61]   Does the architecture, design, or code create any maintenance difficulties?

[62]   Are the maintenance people involved early in the design?

[63]   Is the product documentation adequate for maintenance by an outside organization?

b)   Reliability
[Are the reliability or availability requirements difficult to meet?]

[64]   Are reliability requirements allocated to the software?

[65]   Are availability requirements allocated to the software?

   (Yes)   (65.a) Are recovery timelines any problem?

c)   Safety
[Are the safety requirements infeasible and not demonstrable?]

[66]   Are safety requirements allocated to the software?

   (Yes)   (66.a) Do you see any difficulty in meeting the safety requirements?

[67]   Will it be difficult to verify satisfaction of safety requirements?

d) Security
[Are the security requirements more stringent than the current state of the practice or program experience?]

[68] Are there unprecedented or state-of-the-art security requirements?

[69] Is it an Orange Book system?

[70] Have you implemented this level of security before?

e) Human Factors
[Will the system will be difficult to use because of poor human interface definition?]

[71] Do you see any difficulty in meeting the Human Factors requirements?

(No) (71.a) How are you ensuring that you will meet the human interface requirements?

If prototyping

(Yes) (71.a.1) Is it a throw-away prototype?

(No) (71a.2) Are you doing evolutionary development?

(Yes)(71.b) Are you experienced in this type of development?

(Yes)(71.c) Are interim versions deliverable?

(Yes)(71.d) Does this complicate change control?

f) Specifications
[Is the documentation adequate to design, implement, and test the system?]

[72] Is the software requirements specification adequate to design the system?

[73] Are the hardware specifications adequate to design and implement the software?

[74] Are the external interface requirements well specified?

[75] Are the test specifications adequate to fully test the system?

If in or past implementation phase

[76] Are the design specifications adequate to implement the system?

• Internal interfaces

B. Development Environment

6. Development Process

a) <u>Formality</u>
[Will the implementation difficult to understand or maintain?]

[77] Is there more than one development model being used?

  - Spiral
  - Waterfall
  - Incremental

  (Yes) (77.a) Is coordination between them a problem?

[78] Are there formal, controlled plans for all development activities?

  - Requirements analysis
  - Design
  - Code
  - Integration and test
  - Installation
  - Quality assurance
  - Configuration management

  (Yes) (78.a) Do the plans specify the process well?

  (Yes) (78.b) Are developers familiar with the plans?

b) Suitability
[Is the process suited to the development model, e.g., spiral, prototyping?]

[79] Is the development process adequate for this product?

[80] Is the development process supported by a compatible set of procedures, methods and tools?

c) Process Control
[Is the software development process enforced, monitored and controlled using metrics? Are distributed development sites coordinated?]

[81] Does everyone follow the development process?

(Yes) (81.a) How is this insured?

[82] Can you measure whether the development process is meeting your productivity and quality goals?

If there are distributed development sites

[83] Is there adequate coordination among distributed development sites?

d) Familiarity
[Are the project members experienced in use of the process? Is the process understood by all staff members?]

[84] Are people comfortable with the development process?

e) Product Control
[Are there mechanisms for controlling changes in the product?]

[85] Is there a requirements traceability mechanism that tracks requirements from the source specification through test cases?

[86] Is the traceability mechanism used in evaluating requirement change impact analyses?

[87] Is there a formal change control process?

(Yes) (87.a) Does it cover all changes to base-lined requirements, design, code, and documentation?

[88] Are changes at any level mapped up to the system level and down through the test level?

[89] Is there adequate analysis when new requirements are added to the system?

[90] Do you have a way to track interfaces?

[91] Are the test plans and procedures updated as part of the change process?

7. Development System

a) Capacity
[Is there sufficient work station processing power, memory, or storage capacity?]

[92] Are there enough workstations and processing capacity for all staff?

[93] Is there sufficient capacity for overlapping phases, such as coding, integration and test?

b) Suitability
[Does the development system support all phases, activities, and functions?]

[94] Does the development system support all aspects of the program?

- Requirements analysis
- Performance analysis
- Design
- Coding
- Test
- Documentation

- Configuration management
- Management tracking
- Requirements traceability

c)   Usability
[How easy is the development system to use?]

[95]   Do people find the development system easy to use?

[96]   Is there good documentation of the development system?

d)   Familiarity
[Little prior company or project member experience with the development system?]

[97]   Have people used these tools and methods before?

e)   Reliability
[Does the system suffer from software bugs, down-time, insufficient built-in back-up?]

[98]   Is the system considered reliable?

- Compiler
- Development tools
- Hardware

f)   System Support
[Is there timely expert or vendor support for system?]

[99]   Are the people trained in use of the development tools?

[100]   Do you have access to experts in use of the system?

[101]   Do the vendors respond to problems rapidly?

g) Deliverability

[Are the definition and acceptance requirements defined for delivering the development system to the customer; not budgeted? *HINT: if the participants are confused about this, it is probably not an issue from a risk perspective.*]

[102] Are you delivering the development system to the customer?

(Yes) (102.a) Have adequate budget, schedule, and resources been allocated for this deliverable?

8. Management Process

a) Planning

[Is the planning timely technical leads included contingency planning done?]

[103] Is the program managed according to the plan?

(Yes) (103.a) Do people routinely get pulled away to fight fires?

[104] Is re-planning done when disruptions occur?

[105] Are people at all levels included in the planning of their own work?

[106] Are there contingency plans for known risks?

(Yes) (106.a) How do you determine when to activate the contingencies?

[107] Are long-term issues being adequately addressed?

b) Project Organization

[Are the roles and reporting relationships clear?]

[108] Is the program organization effective?

[109] Do people understand their own and others roles in the program?

[110] Do people know who has authority for what?

c) Management Experience

[Are the managers experienced in software development, software management, the application domain, the development process, or on large programs?]

[111] Does the program have experienced managers?

> - Software management
> - Hands-on software development
> - With this development process
> - In the application domain
> - Program size or complexity

d) Program Interfaces

[Poor interface with customer, other contractors, senior and/or peer managers.]

[112] Does management communicate problems up and down the line?

[113] Are conflicts with the customer documented and resolved in a timely manner?

[114] Does management involve appropriate program members in meetings with the customer?

> - Technical leaders
> - Developers
> - Analysts

[115] Does management work to ensure that all customer factions are represented in decisions regarding functionality and operation?

[116] Does the project always present an optimistic picture to the customer or senior management?

9. Management Methods

a) Monitoring
[Are management metrics defined; development progress tracked?]

[117] Are there periodic structured status reports?

(Yes) (117.a) Do people get a response to their status reports?

[118] Does appropriate information get reported to the right organizational levels?

[119] Do you track progress versus plan?

(Yes) (119.a) Does management have a clear picture of what is going on?

b) Personnel Management
[Are project personnel trained and used appropriately?]

[120] Do people get trained in skills required for this program?

(Yes) (120.a) Is this part of the program plan?

[121] Do people get assigned to the program who do not match the experience profile for your work area?

[122] Is it easy for program members to get management action?

[123] Are program members at all levels aware of their status versus plan?

[124] Do people assiduously keep to the plan?

[125] Does management consult with people before making decisions that affect their work?

[126] Does program management involve appropriate program members in meetings with the customer?

  • Technical leaders
  • Developers
  • Analysts

c) **Quality Assurance**
[Are there adequate procedures and resources to assure product quality?]

[127] Is the Software Quality Assurance function adequately staffed on this program?

[128] Do you have defined mechanisms for assuring quality?

(Yes) (128.a) Do all areas and phases have quality procedures?

(Yes) (128.b) Are people used to working with these procedures?

d) **Configuration Management**
[Are the change procedures or version control, including installation site(s) adequate?]

[129] Do you have an adequate configuration management system?

[130] Is the Configuration Management function adequately staffed?

[131] Is coordination required with an installed system?

(Yes) (131.a) Is there adequate configuration management of the installed system?

(Yes) (131.b) Does the configuration management system synchronize your work with site changes?

[132] Are you installing in multiple sites?

(Yes) (132.a) Does the configuration management system provide for multiple sites?

5. Work Environment

a) Quality Attitude
[Is there a lack of orientation toward quality workmanship?]

[133] Are all staff levels oriented toward quality procedures?

[134] Does schedule get in the way of quality?

b) Cooperation
[Is there a lack of team spirit; does conflict resolution require management intervention?]

[135] Do people work cooperatively across functional boundaries?

[136] Do people work effectively towards common goals?

[137] Is management intervention sometimes required to get people working together?

c) Communication
[Is there poor awareness of mission or goals; poor communication of technical information among peers and managers?]

[138] Is there good communication among the members of the program?

- Managers
- Technical leaders
- Developers

- Testers
- Configuration management
- Quality assurance

[139] Are the managers receptive to communication from program staff?

(Yes) (139.a) Do you feel free to ask your managers for help?

(Yes) (139.b) Are members of the program able to raise risks without having a solution in hand?

[140] Do the program members get timely notification of events which may affect their work?

(Yes) (140.a) Is this formal or informal?

d) Morale

[Is there a non-productive, non-creative atmosphere? Do people feel that there is no recognition or reward for superior work?]

[141] How is morale on the program?

(No) (141.a) What is the main contributing factor to low morale?

[142] Is there any problem keeping the people you need?

C. Program Constraints

1. Resources

a) Schedule
[Is the schedule inadequate or unstable?]

[143] Has the schedule been stable?

[144] Is the schedule realistic?

   (Yes) (144.a) Is the estimation method based on historical data?

   (Yes) (144.b) Has the method worked well in the past?

[145] Is there anything for which adequate schedule was not planned?

   - Analysis and studies
   - QA
   - Training
   - Maintenance courses and training
   - Capital equipment
   - Deliverable development system

[146] Are there external dependencies which are likely to impact the schedule?

b) Staff
[Is the staff inexperienced, lacking domain knowledge, lacking skills, or understaffed?]

[147] Are there any areas where the required technical skills are lacking?

   - Software engineering and requirements analysis method

- Algorithm expertise
- Design and design methods
- Programming languages
- Integration and test methods
- Reliability
- Maintainability
- Availability
- Human factors
- Configuration management
- Quality assurance
- Target environment
- Level of security
- COTS
- Reuse software
- Operating system
- Database
- Application domain
- Performance analysis
- Time-critical applications

[148] Do you have adequate personnel to staff the program?

[149] Is the staffing stable?

[150] Do you have access to the right people when you need them?

[151] Have the program members implemented similar systems of this type?

[152] Is the program reliant on a few key people?

[153] Is there any problem with getting cleared people?

c) Budget
[Is the funding insufficient or unstable?]

[154] Is the budget stable?

[155] Is the budget based on a realistic estimate?

(Yes) (155.a) Is the estimation method based on historical data?

(Yes) (155.b) Has the method worked well in the past?

[156] Have features or functions been deleted as part of a design-to-cost effort?

[157] Is there anything for which adequate budget was not allocated?

- Analysis and studies
- QA
- Training
- Maintenance courses
- Capital equipment
- Deliverable development system

[158] Do budget changes accompany requirement changes?

(Yes) (158.a) Is this a standard part of the change control process?

d) Facilities
[Are the facilities adequate for building and delivering the product?]

[159] Are the development facilities adequate?

[160] Is the integration environment adequate?

2. Contract

a) Type of Contract
[Is the contract type a source of risk to the program?]

[161] What type of contract do you have? (Cost plus award fee, fixed price,.....)

(161a) Does this present any problems?

[162] Is the contract burdensome in any aspects of the program?

- SOW (Statement of Work)
- Specifications
- Data Item Descriptions
- Contract parts
- Excessive customer involvement

[163] Is the required documentation burdensome?

- Excessive amount
- Picky customer
- Long approval cycle

b) Restrictions
[Does the contract cause any restrictions?]

[164] Are there problems with data rights?

- COTS software
- Developmental software
- Non-developmental Items

c) Dependencies
[Does the program have any dependencies on outside products or services?]

[165] Are there dependencies on external products or services which may affect the product, budget or schedule?

- Associate contractors

- Prime contractor
- Subcontractors
- Vendors or suppliers
- Customer furnished equipment or software

3. Program Interfaces

a) Customer

[Are there any customer problems such as: lengthy document-approval cycle, poor communication, and inadequate domain expertise?]

[166] Is the customer approval cycle timely?

- Documentation
- Program reviews
- Formal reviews

[167] Do you ever proceed before receiving customer approval?

[168] Does the customer understand the technical aspects of the system?

[169] Does the customer understand software?

[170] Does the customer interfere with process or people?

[171] Does management work with the customer to reach mutually agreeable decisions in a timely manner?

- Requirements understanding
- Test criteria
- Schedule adjustments
- Interfaces

[172] How effective are your mechanisms for reaching agreements with the customer?

- Working groups (Contractual?)
- Technical Interchange Meetings (Contractual?)

[173] Are all customer factions involved in reaching agreements?

(Yes) (173.a) Is it a formally defined process?

[174] Does management present a realistic or optimistic picture to the customer?

If there are associate contractors

b)   Associate Contractors
[Are there any problems with associate contractors such as inadequately defined or unstable interfaces, poor communication or cooperation?]

[175] Are the external interfaces changing without adequate notification, coordination or formal change procedures?

[176] Is there an adequate transition plan?

(Yes) (176.a) Is it supported by all contractors and site personnel?

[177] Is there any problem with getting schedules or interface data from associate contractors?

(No) (177.a) Are they accurate?

If there are sub-contractors

c)   Subcontractors
[Is the program dependent on subcontractors for any critical areas?]

[178] Are there any ambiguities in subcontractor task definitions?

[179] Is the subcontractor reporting and monitoring procedure different from the program's reporting requirements?

[180] Is subcontractor administration and technical management done by a separate organization?

[181] Are you highly dependent on subcontractor expertise in any areas?

[182] Is subcontractor knowledge being transferred to the company?

[183] Is there any problem with getting schedules or interface data from subcontractors?

If program is a sub-contract

d) Prime Contractor
[Is the program facing difficulties with its Prime contractor?]

[184] Are your task definitions from the Prime ambiguous?

[185] Do you interface with two separate prime organizations for administration and technical management?

[186] Are you highly dependent on Prime's expertise in any areas?

[187] Is there any problem with getting schedules or interface data from the Prime?

e) Corporate Management
[Is there a lack of support or micro management from upper management?]

[188] Does program management communicate problems to senior management?

(Yes) (188.a) Does this seem to be effective?

[189] Does corporate management give you timely support in solving your problems?

[190] Does corporate management tend to micro-manage?

[191] Does management present a realistic or optimistic picture to senior management?

f)   Vendors
[Are vendors responsive to programs needs?]

[192] Are you relying on vendors for deliveries of critical components?

- Compilers
- Hardware
- COTS

g)   Politics
[Are politics causing a problem for the program?]

[193] Are politics affecting the program?

- Company
- Customer
- Associate contractors
- Subcontractors

[194] Are politics affecting technical decisions?

Glossary

Application domain – A term that refers to the nature of the application. Examples are real-time flight control systems and management information systems.

Availability – The relative time that an operational product must be available for use. Usually expressed as the ratio of time available for use to some total time period or as specific hours of operation.

Change control – A part of configuration management that reviews, approves, and tracks progress of alterations in the configuration of a configuration item delivered, to be delivered, or under formal development, after formally establishing its configuration identification.

Configuration management – A discipline applying technical and administrative direction and surveillance to identify and document the functional and physical characteristics of a controlled item, control changes to a configuration item and its documentation, and record and report change processing and implementation status.

Configuration management system – The processes, procedures, and tools used by the development organization to accomplish configuration management.

COTS (commercial off-the-shelf) – A type of non-developmental software that is supplied by commercial sources.

Customer – The person or organization receiving a product or service. There may be many different customers for individual organizations within a program structure. Government program offices may view the customer as the user organization for which they are managing the project. Contractors may view the program office as well as the user organization as customers.

Design specifications – A document that prescribes the form, parts, and details of the product according to a plan.

Design-to-cost – A term describing the bidding of a selected, reduced set of requirements to meet cost objectives.

Development computer – The hardware and supporting software system used for software development.

Development facilities – The office space, furnishings, and equipment that support the development staff.

Development model – The abstract visualization of how the software development functions (such as requirements definition, design, code, test, and implementation) are organized. Typical models are the waterfall model, the iterative model, and the spiral model.

Development process – The implemented process for managing the development of the deliverable product. For software, the development process includes the following major activities, which may overlap and may be applied iteratively or recursively:

- System Requirements Analysis/Design
- Software Requirements Analysis
- Preliminary Design
- Detailed Design
- Coding and Computer Software Unit Testing
- Computer Software Component Integration and Testing
- Computer Software Configuration Item Testing
- System Integration and Testing

Development sites – The locations at which development work is being conducted.

Development system – The hardware and software tools and supporting equipment that will be used in product development including such items as computer aided software engineering (CASE) tools, compilers, and configuration management systems.

External dependencies – Any deliverable product or service from other organizations that is critical to the project or program.

External interfaces – The points where the software system under development interacts with other systems, sites, or people.

Hardware specifications – A document that prescribes the functions, materials, dimensions, and workmanship that a hardware item must meet.

Implementation – The act of preparing the product for use by the customer.

Integration – The act of assembling individual hardware and/or software components into a usable whole.

Integration environment – The hardware, software, and supporting tools that will be used to support product integration.

Internal interfaces – The points where the software system under development interacts with other components of the system under development.

Long-term issues – Issues of strategic importance to the project that can be compromised in the heat of battle. Issues such as employee training and development, establishing and improving processes and procedures, and similar activities are important to the long-term viability of the project and the organization.

Non-developmental software (NDS) – Deliverable software that is not developed under the contract but is provided by the contractor, the government, or a third party. NDS may be referred to as reusable software, government-furnished software, or commercially available software, depending on its source.

Orange Book – A security standard set by the U.S. government.

Product integration – The act of assembling individual hardware and software components into a functional whole.

Re-engineering – The practice of adapting existing software artifacts or systems to perform new or enhanced functions. Re-engineered artifacts may be substantially different from existing artifacts.

Reliability – The degree of dependability that an operational product must meet. Usually expressed as the average time to failure.

Reusing – Hardware or software developed in response to the requirements of one application that can be used, in whole or in part, to satisfy the requirements of another application.

Safety – The degree to which the software product minimizes the potential for hazardous conditions during its operational mission.

Security – The degree to which a software product is safe from unauthorized use.

Software requirements specification – A document that contains the complete set of engineering requirements for each computer software configuration item.

System integration – The activities of assembling hardware and software components into a deliverable product.

Target computer – The hardware and supporting software system that will actually be used when the software system is fielded.

TBDs – to be defined.

Test specifications – A document that prescribes the process and procedures to be used to verify that a product meets its requirements.

Traceability mechanism – Processes and procedures (manual and/or automated) that map all software components and artifacts from source requirements through test cases.

Transition plan – A plan (documented in the Computer Resources Integrated Support document) specifying how products are to transition from development to support.

Unit testing – Testing performed by the developers on an individual software or hardware component to verify the component meets its allocated requirements.

Index

technical
    basis 3, 9
    consensus 44
    leaders 45
tool support 63
training 3, 19, 34, 43, 50
types of functions 19

V

validation 5, 6, 19, 33, 45
validity of results 33
verification 17, 33, 45
versions 4, 5

W

wording 28

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION	1b. RESTRICTIVE MARKINGS
Unclassified	None

2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT
N/A	Approved for Public Release
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	Distribution Unlimited
N/A	

4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
CMU/SEI-94-TR-19	ESC-TR-94-019

6a. NAME OF PERFORMING ORGANIZATION	6b. OFFICE SYMBOL (if applicable)	7a. NAME OF MONITORING ORGANIZATION
Software Engineering Institute	SEI	SEI Joint Program Office

6c. ADDRESS (city, state, and zip code)	7b. ADDRESS (city, state, and zip code)
Carnegie Mellon University Pittsburgh PA 15213	HQ ESC/ENS 5 Eglin Street Hanscom AFB, MA 01731-2116

8a. NAME OFFUNDING/SPONSORING ORGANIZATION	8b. OFFICE SYMBOL (if applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
SEI Joint Program Office	ESC/ENS	F1962890C0003

8c. ADDRESS (city, state, and zip code))	10. SOURCE OF FUNDING NOS.			
Carnegie Mellon University Pittsburgh PA 15213	PROGRAM ELEMENT NO	PROJECT NO.	TASK NO	WORK UNIT NO.
	63756E	N/A	N/A	N/A

11. TITLE (Include Security Classification)
Software Risk Evaluation Method Version 1.0

12. PERSONAL AUTHOR(S)
Frank J. Sisti, Sujoe Joseph

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (year, month, day)	15. PAGE COUNT
Final	FROM TO	December 1994	191

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (continue on reverse of necessary and identify by block number)
FIELD	GROUP	SUB. GR.	software risk evaluation
			software risks
			software acquisitions

19. ABSTRACT (continue on reverse if necessary and identify by block number)

The Software Risk Evaluation (SRE) method is used for identifying, analyzing, communicating, and mitigating software risks.

The SRE method is intended to be used by decision-makers for managing the software risks of software-intensive programs and projects. The SRE method facilitates the mitigation of software risks for managers.

This document reports on the Software Risk Evaluation method version 1.0 and provides a high-level description of the method. The report is intended to provide the reader with an overview of the functional components and the implementation aspects of the method.

The current version of the SRE method evolved from two earlier versions. The first version was developed under a technical collaboration agreement with the Director of Defense Research and Engineering for mitigating the risks inherent in software acquisitions. The latter version was developed to improve and enhance the capabilities of the method for users in software-intensive programs and projects.

(please turn over)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT			21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED/UNLIMITED ■	SAME AS RPT □	DTIC USERS ■	Unclassified, Unlimited Distribution

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (include area code)	22c. OFFICE SYMBOL
Thomas R. Miller, Lt Col, USAF	(412) 268-7631	ESC/ENS (SEI)